

**LEBANESE AMERICAN UNIVERSITY**

Query Acceleration in Multimedia Database Systems

By

Rawa Karaki

A thesis

Submitted in partial fulfillment of the requirements  
For the degree of Master of Science in Computer Science

School of Arts and Sciences  
June 2014



Lebanese American University

School of Arts and Sciences - Beirut Campus

## THESIS APPROVAL FORM

Student Name: Rawa Karaki

I.D. #: 201003638

Thesis Title : Accelerating Queries in Multimedia Databases

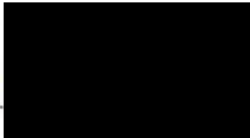
Program: Masters of Science in Computer Science


Department: Computer Science and Mathematics


School: Arts and Sciences

The undersigned certify that they have examined the final electronic copy of this thesis and approved it in Partial Fulfillment of the requirements for the degree of:

Masters of Science in the major of Computer Science

Thesis Advisor's Name: Ramzi A. Haraty Signature:  Date: June 4, 2014

Committee Member's Name: Samer Habre Signature:  Date: June 4, 2014


Committee Member's Name: Bechara Al Bouna Signature:  Date: June 4, 2014

**THESIS COPYRIGHT RELEASE FORM**

LEBANESE AMERICAN UNIVERSITY NON-EXCLUSIVE DISTRIBUTION LICENSE

By signing and submitting this license, you (the author(s) or copyright owner) grants to Lebanese American University (LAU) the non-exclusive right to reproduce, translate (as defined below), and/or distribute your submission (including the abstract) worldwide in print and electronic format and in any medium, including but not limited to audio or video. You agree that LAU may, without changing the content, translate the submission to any medium or format for the purpose of preservation. You also agree that LAU may keep more than one copy of this submission for purposes of security, backup and preservation. You represent that the submission is your original work, and that you have the right to grant the rights contained in this license. You also represent that your submission does not, to the best of your knowledge, infringe upon anyone's copyright. If the submission contains material for which you do not hold copyright, you represent that you have obtained the unrestricted permission of the copyright owner to grant LAU the rights required by this license, and that such third-party owned material is clearly identified and acknowledged within the text or content of the submission. IF THE SUBMISSION IS BASED UPON WORK THAT HAS BEEN SPONSORED OR SUPPORTED BY AN AGENCY OR ORGANIZATION OTHER THAN LAU, YOU REPRESENT THAT YOU HAVE FULFILLED ANY RIGHT OF REVIEW OR OTHER OBLIGATIONS REQUIRED BY SUCH CONTRACT OR AGREEMENT. LAU will clearly identify your name(s) as the author(s) or owner(s) of the submission, and will not make any alteration, other than as allowed by this license, to your submission.

Name: *Rawa Karaki*

Signature: 


Date: *4/6/2014*

**PLAGIARISM POLICY COMPLIANCE STATEMENT**

I certify that:

- I have read and understood LAU's Plagiarism Policy.
- I understand that failure to comply with this Policy can lead to academic and disciplinary actions against me.
- This work is substantially my own, and to the extent that any part of this work is not my own I have indicated that by acknowledging its sources.

Name: *Rawa Karaki*

Signature: 

Date: *04/06/2014*

## ACKNOWLEDGMENT

Foremost, I would like to express my sincere gratitude to my advisor Dr. Ramzi Haraty for the continuous support of my study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor.

I would also like to thank my family: my parents, my sister and my husband for supporting me spiritually throughout my life.

## Query Acceleration in Multimedia Database Systems

Rawa Karaki

### ABSTRACT

With the increasing popularity of the World Wide Web comes the enormous increase in stored digital contents, which could challenge users to search and use the multimedia data efficiently. This work focuses on hastening techniques for efficient retrieval of multimedia data. In this thesis, we exploit the use of bit-vectors to accelerate queries in multimedia databases. We also use a compressed bit-vector to minimize the amount of data cached on disk; thus reducing the amount of memory and time needed to execute queries. We also compare our scheme with other related strategies.

Keywords: Multimedia, Database, Retrieval, Compressed, Bit-Vector, Metadata, Objects, Files, Query.

# Table of Contents

Chapter	Page
<b>Introduction .....</b>	<b>1</b>
1.1 What is Multimedia Database?.....	1
1.2 Types of Multimedia Data.....	4
1.3 Multimedia Database Applications.....	5
1.4 Problem statement: .....	6
1.5 Thesis Organization:.....	8
<b>II- Background &amp; Related Work.....</b>	<b>9</b>
2.1 Introduction .....	9
2.2. Retrieval by Browsing: .....	10
2.2.1 Comparing Browsing Model to Query Model.....	11
2.3. Retrieval by Metadata Attributes .....	13
2.3.1 Metadata Classification .....	18
2.3.2 Source of Metadata .....	18
2.4. Retrieval by Shape Similarity .....	19
2.4.1 Example.....	19
2.4.2 Edge thinning:.....	22
2.4.3 Multimedia Shape Retrieval Classification .....	23
2.4.4 Multimedia Shape Retrieval Tool.....	25
2.5. Content Base Retrieval.....	28
2.5.1 Content Base Retrieval for Images .....	30
2.5.2 Content Base Retrieval for Video.....	31
2.5.3 Content Base Retrieval for Audio .....	33
<b>III- Using Compressed Bit-Vector for Multimedia Data Retrieval.....</b>	<b>36</b>
3.1 Introduction .....	36
3.2 Algorithm Details .....	39

3.3 Experimental results: .....	43
3.4 Conclusion:.....	56
<b>IV- Conclusion .....</b>	<b>58</b>
4.1 Conclusion.....	58
4.2 Future Work.....	59
<b>References .....</b>	<b>60</b>



# List of Figures

FIGURE 1.THE INFORMEDIA ARCHITECTURE.[28] .....	3
FIGURE 2. QUERY AS A FILTER [27] .....	12
FIGURE 3. BROWSING AS EXAMINATION OF STRUCTURE [27] .....	13
FIGURE 4. “METADATA EXAMPLE OF AN IMAGE”.[30] .....	14
FIGURE 5.VAN GOGH’S WHEAT FIELD UNDER THREATENING SKIES (A) AND THE SKETCH OF A QUERY TO RETRIEVE IT (B)[31].....	20
FIGURE 6.OVERALL ARCHITECTURE OF OBJECT EXTRACTOR.[14] .....	26
FIGURE 7.FLOOD LL FOR EXTRACTION (FFE) ALGORITHM .....	27
FIGURE 8.CONTENT BASED VIDEO RETRIEVAL SYSTEMS.[5].....	32
FIGURE 9.ALGORITHM WORK FLOW .....	39
FIGURE 10 .BIT VECTOR TABLE .....	43
FIGURE 11 .QUERY EXAMPLE WITH PROCESSING TIME.....	44
FIGURE 12 .CALL OF THE STORED PROCEDURE.....	44
FIGURE 13 . SECOND CALL OF STORED PROCEDURE.....	45
FIGURE 14. PHP PROCESS RETURNING THE FINAL RESULT .....	46
FIGURE 15.BIT VECTOR BUILDING TIME .....	48
FIGURE 16.SECOND BIT VECTOR BUILDING TIME .....	48
FIGURE 17. PHP ALGORITHM.....	54
FIGURE 18. PERFORMANCE ANALYSIS FOR A SIMPLE QUERY IN MULTIMEDIA DATABASE.....	55
FIGURE 19 . PERFORMANCE ANALYSIS FOR INNER JOIN QUERY IN MULTIMEDIA DATABASE .....	56

# List of Tables

TABLE 1. SAMPLE MEDIA TYPES, FORMATS, AND RELATED DATA VOLUMES AND TRANSFER RATES [29] .....	7
TABLE 2. RUNNING TIME FOR DIFFERENT QUERIES .....	49
TABLE 3. SECOND RUNNING TIME FOR DIFFERENT QUERIES .....	54

# Chapter One

## Introduction

### 1.1 What is Multimedia Database?

Multimedia databases have become one of the puffs in Computer Science technology. It is a recent evolution of the Internet and data warehousing, with countless books and articles showing interest in the field. Many authors wrote about the evolution of multimedia databases and ways to implement it. Multimedia is a mix of multiple mediums - images, sounds, music, audios and videos etc. We use and handle media files on a daily basis, and they are included in many applications such as artwork, teaching, schooling, training, medical science, advertisements, and technical research. There are two types of multimedia files--static and dynamic media. Static media are time-independent media like text, graphics, and images; while dynamic media are time-dependent (media that moves over time) objects like images, audio, and video. As long as the development of the Internet and computer technology continues, multimedia files will appear more and more in many applications. Multimedia will influence our existence. For that reason, it is important and significant that the data files of multimedia objects are arranged, ordered and categorized so we can simply access them at any time. Therefore,

multimedia databases are the necessary tool to handle and support these enormous multimedia object files.

A multimedia database is a type of database that is similar to all other database types except that it contains multimedia files in its collection. To organize and manage multimedia data files, a multimedia database management system is needed. It is a program that runs and directs the collection of media files and allows entry for end users to retrieve multi-media file or objects. In general, multimedia databases hold images, audio, video, animations and many other file forms. But, all files or data are saved in binary form in the multimedia database. Figure 1 shows that the multimedia software supports two operations: The creation and exploration of multimedia Databases

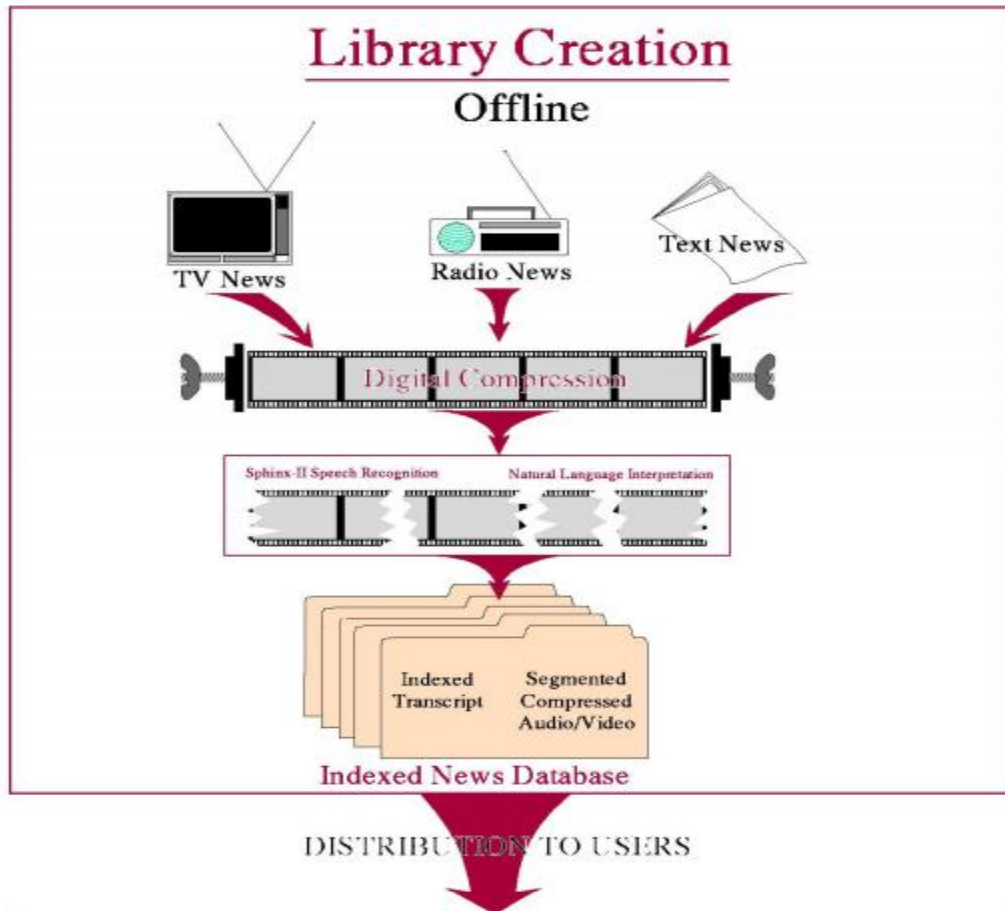


Figure 1: The informedia architecture. [28]

## 1.2 Types of Multimedia Data

Data types can be characterized as a number of different data types. The different basic types of multimedia data type are reported as follows:

- **Text:** Text can be saved in many different forms. Texts are stored in the database as a multimedia objects. Text fonts can vary allowing a more complex structure. Colors, shades, bold letters can be added to any text before saving it in multimedia databases.
- **Images:** An image is a collection of pixels that illustrate a division in the end user's graphical presentation. The image size differs from one to image to another according to its size, resolution, twist, complication, compact and cut used to cache images. Well-known images formats are bmp, gif and jpg.
- **Audio:** Generated from an aural recording device, an audio file is a well-known data category being merged in most of users' applications. Many techniques are used to compress audio files because they consume large space. A 1-minute recording can take up to 3 MBs of space, for example.
- **Video:** a video file is a series of pictures (called sequence of segmentation), which records and documents a real-life phenomenon and is produced usually by a videotape keeper. It is the most space consuming of all multimedia data types. It also depends on the design, compression and area of a particular framework. A single frame can be 1 MB when saved in a file.

- **Graphic Objects:** Graphic Objects are special data structures that define 2D objects as ordinary drawings, sketches, and illustrations, or 3D items. These include multiple styles used by pictures or video software.

### **1.3 Multimedia Database Applications**

Multimedia database implementation differs from regular database implementation in the design of the media objects and files where the files are kept and stored. Different characteristics of multimedia data represent the diversity of the data since they are complex--composed of audio-visual data. Research shows that objects in multimedia data are complex and involve a chained structure that can hold a connection between them. Static media are time-independent like text, graphics, and images. For instance, image files do not have time-related action because there is no connected time factor. Video files, on the other, are dynamic, and have both time and dimensional dependency. This is because the video is composed of multiple ordered image frames which combine to form the video file.

Several implementations of multimedia databases are:

1. Documenting and keeping records
2. Distributing knowledge
3. Educating and Training
4. Marketing, Advertising, Entertaining, Traveling
5. Monitoring and real-time Control

## **1.4 Problem statement:**

With the evolution of Internet and computer users, multimedia structure has a greater effect on our daily life. That is why finding a new technique to easily retrieve enormous multimedia information and files, at any point of time, is in high demand. Any multimedia object can be generally described as a group of extended, shapeless series of bytes. These objects are called BLOBs: Binary Large Objects. BLOB files are usually very large in size, for this reason, database management systems provide particular maintenance to insert, delete, modify or retrieve BLOB object from database.

Modern databases are frequently capable of storing BLOBs and CLOBs, Binary Large Objects and Character Large Objects respectively, as columns in their tables. Data stored in a BLOB column can be accessed using connectors and manipulated using client-side code. Reading a BLOB from the database is a slow task considering the size of a multimedia object. A BLOB can contain as much as 4 gigabytes of data for each field. Multimedia database systems are thus required to provide an efficient cache of the BLOB files, but this is not sufficient for multimedia implementation maintenance. Therefore, a query of a prolonged continual series of bytes is restricted to a matching pattern and reorganization of a BLOB multimedia object may return zero results due to missing constructional information. Even if it can be realized, to draw out information of the object in realistic time, for example working with pattern identification techniques, would be unrealistic. For that reason, a multimedia database system should keep an analytical structure of the BLOB files. Multimedia objects can be saved in smaller parts to allow easier retrieval of BLOB objects based on content. Multimedia data is sizeable and have an impact on the retrieval, insertion and manipulation of multimedia data files. The large



amounts of data to be processed can be checked against those that need to be processed.

Table 1 illustrates the enormous sizes of data for media file of different types.

**Table 1: Sample Media Types, formats, and related data volumes and transfer rates [29]**

<b>Media Type</b>	<b>Sample Format</b>	<b>Data Volume</b>	<b>Transfer Rate</b>
Text	ASCII	1MB/ 500 pages	2KB/page
B/W Image	G3/4-Fax	32MB/500 images	64KB/page
-Color Image	GIF,TIFF;JPEG	1.6GB/500 images 0.2GB/500 images	3.2MB/image 0.4MB/image
CD-music	CD-DA	52.8MB/5 minutes	176KB/sec.
Consumer Video	PAL	6.6GB/5 minutes	22MB/sec.
High quality video	HDTV	33GB/5 minutes	110MB/sec.
Speech	m-law,linear; ADPCM,PEG audio	2.4 MB/5 minutes 0.6MB, 0.2MB/5 min.	8KB/sec.

Similar to the matching problems stated above, we need to handle the enormous number of media data files with real time limitations. This seriously affects the design of the network, software, and hardware. These constrains must be taken into consideration when building any database system that handle multimedia objects.

Not all user queries can return answers in multimedia databases and may often return inexact answers. The response to a multimedia query can be a complicated multimedia disposition for the user to explore [4]. Many works have focused on returning efficient answers to user queries but we still do not have real methods that return exact media matching.

Our algorithm use a compress bit vector for multimedia data retrieval to fast select files from the database. The method facilitates rapid searching of multimedia data objects in a multimedia database. A single bit vector is used to determine matches for the main query, returning a reduced set of multimedia objects instead of the entire multimedia data object, thereby greatly reducing the query search time, increasing the efficiency of the process by allowing the bit-level operations and minimizing the cost and amount of data transferred. The execution time is exactly proportional to the size of input. The algorithm complexity is of order  $O(n)$ .

### **1.5 Thesis Organization:**

The remainder of the thesis is arranged as follow:

In Chapter 2, we demonstrate related works made in the area of Multimedia Databases.

In Chapter 3, we give an explanation and description of the algorithm: compressed bit vector for multimedia data retrieval. The algorithm will be tested and evaluated on real data compared with the existing methods.

In Chapter 4, we summarize the contributions and achievements of this thesis, summarize concluding remarks, declare the future work plans, and submit the list of publications derived from this thesis.

# Chapter Two

## Background & Related Work

### 2.1 Introduction

Querying and retrieving information in multimedia databases differs from traditional databases [8]. A fairly straightforward search can be done in alphanumeric databases. Multimedia databases contain pictures and different complex multimedia data objects, thus the database is not easily indexed, classified and retrieved [16]. How is it possible to retrieve a picture with a cup of water or a horoscope sign? Those shapes are difficult to recognize. Some retrieval classes for Multimedia Databases include:

- Retrieval by Browsing (RBR): Browsing multimedia objects to retrieve the best matching file. For example, using a simple interface to let users browse small images known as “thumbnails” to pick the image that matches the query.
- Retrieval by Metadata Attributes (RMA): Designing a query that addresses the Meta and logical characteristics. For this purpose, any media file is stored with information describing the file. For example, we will not query an image with a bird but we will address our search to find which media handle the keyword ‘bird’ as its Meta information.

- Retrieval by Shape Similarity (RSS): It is a type of retrieval based on media content. Searching in a multimedia database based on shape similarity of the file. For example, retrieve all the images that contain a circle.
- Retrieval by Content Attributes (RCA): Query is sent with a detail describing the file to be retrieved. For example, retrieval of all images that contain a specific celebrity.

## **2.2. Retrieval by Browsing:**

A user who requests the search for a specific file uses terms and details to illustrate the retrieval system. Then, the software matches the query with existing matching objects and returns a list of files to the end user for examination. The end user then considers the retrieved files and picks items that exactly match his needs. This type of retrieval works best in finding the exact requested file, but multiple problems appear with its implementation:

1. End users find it hard to formulate queries
2. Queries may return only unwanted files and result in too many suggested unwanted matches
3. Query terms are not properly valued
4. Multiple forms of image and audio files that need conversion

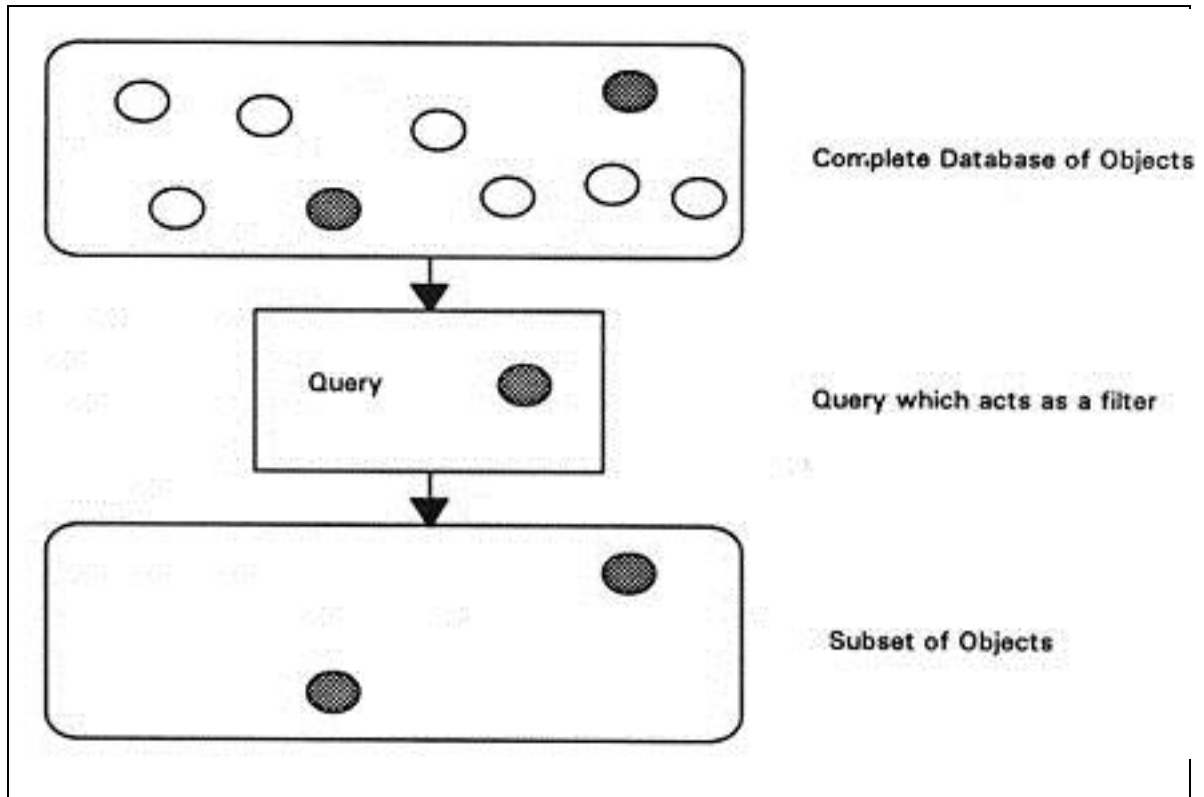
Different authors have proposed that browsing, which uses the human recognition capabilities, can control and solve the above difficulties. Though, even the retrieval by browsing is suggested to be a direction solving many problems in multimedia retrieval

and handling multimedia systems, but it is logically seen as difficult and time inefficient task for humans to solve.

### **2.2.1 Comparing Browsing Model to Query Model**

The browsing model is described as a dependent model that interacts with the end user. In this model, the end user determines the direction of the search and handles the output result from the system being browsed. While retrieval by browsing is an important action of media searching in many systems, it requires time and effort from the end user to handle it [27, 2].

To explain the browsing system; first users select a subset of objects from a bigger database to start examining. Hundreds or more objects are selected before getting the user's exact request from an unstructured database using the attributes value (Figure 2). Figure 2 is a simplified image describing the filter made in complete database objects. A subset of objects is returned to the user to be examined before moving to the next step of choosing the desired media file.



**Figure 2: Query as a Filter [27]**

Searching a media file using the retrieval by browsing model requires the end user to be placed in the organized database. The end user fetches this database based on the received information (Figure 3). The user will fetch the whole database before focusing on the files of interest. This model differs from the query based model even if the end result looks similar. The browsing using query based model request the dynamic reestablishment of the database based on the formulation of the query. In the browsing model, the database remains unchanged; the user only searches the database by moving around media files.

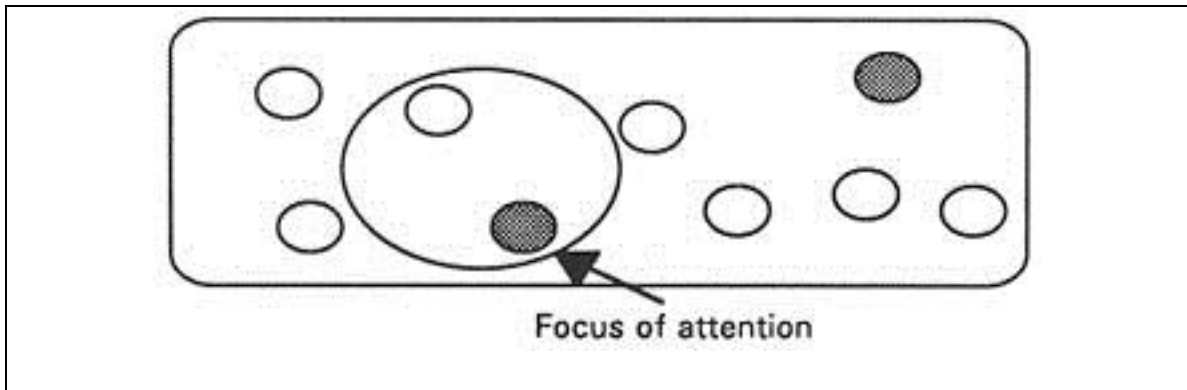


Figure 3: Browsing as Examination of Structure [27]

The requirements of browsing model are as follow:

1. The capability for end users to locate themselves in an area of the database that is of interest.
2. The capability for end users to potentially identify suitable directions in which to develop the search
3. The capability for end users to efficiently and rapidly proceed between the database files.

### **2.3. Retrieval by Metadata Attributes**

Generally, human beings have the power to retrieve and correlate information efficiently. It is unfeasible to search millions of data by simply “staring” in order to assemble diverse documents, which may involve texts, videos, audio and images files, either alone or as multimedia items. Thus, we seek a simple technological multimedia search based on known information of the file.

Metadata are data about data. Metadata can describe any data using different categories: quantity, quality, materials, shape and different properties of the data as tools

to find, understand and access the data files. Metadata details can aid users to have an explanation about the data being searched in multimedia databases. Figure 4 shows a metadata example of the content of the picture file. The picture itself describes nothing than an ordinary image with colors. Without having the metadata description associated with the picture, it will be out of the question for machines to know the properties of this picture. For example, if we would like to know when and where this picture was taken, or its resolution etc., we turn to Metadata. All this information does is provide a key that aids in specifying the properties of the image to be used in many applications [10].

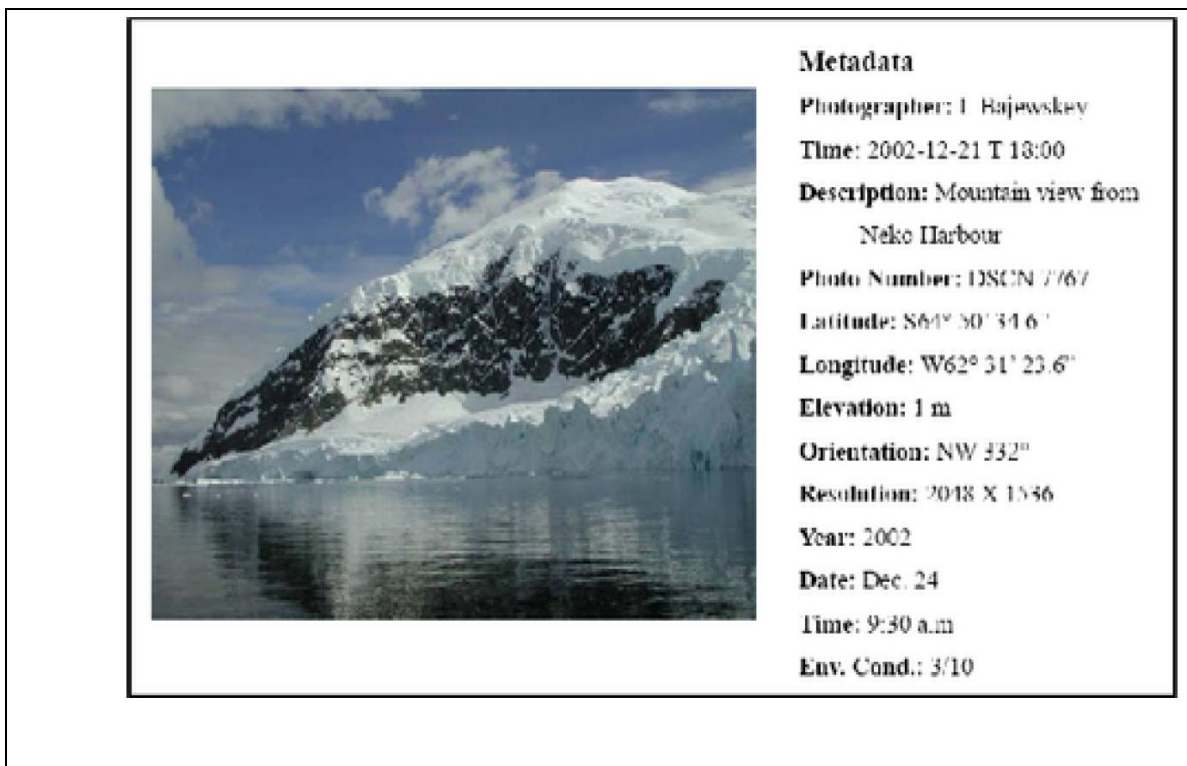


Figure 4: “Metadata example of an image”[30]

In fact, content-based retrieval for multimedia objects is difficult work and returns the same results as the metadata retrieval results. Content-based retrieval is still in its first



phase and could be unused in the near future until completely it becomes efficient. Therefore, the retrieval of multimedia objects should depend on the related information, noted as metadata of different characteristics.

The Metadata model requires descriptive information of the content, combined with contextual information, saved in the multimedia database in reference to the multimedia object, and used as an information tool for browsing search with a point of association of a specific media. Descriptive information is valuable for searching a multimedia object, and is of a major importance when contacting explored results where the attribute, such as the photographer name, singer name or date, are applied to choose and retrieve the file. The metadata representation of the file is flexible and adopts a multilevel approach for describing the file to permit multiple particles to describe the facts and figures of the file. The metadata model may be unusable to work on a single level in describing a media file with multiple classes of representation. For an image, as shown before in Figure 4, multiple descriptive data are associated with saved image snaps that can provide accommodation in the model. For a video file in a broadcasting station, there could be automatically produced information for each shot or segment that describes the scene.

The metadata represents many aspects of the file including content-independent information like the data and time or the location of the file, and content-dependent information like a description of the shape and color of an object inside the image. Those descriptions should give the user the ability to retrieve objects easily and thus, they are the most critical data about the file. To construct the metadata of any file, there are two processes to apply: an application operation and a data operation. These operations are in

different directions: bottom-up and top-down. In a top-down or application operation arrangement, the abstractions and connections motivated by the class of queries for which the associated details in the different media types is handled are relevant. In a bottom-up or data operation arrangement, the metadata is extracted from the data. For this reason, the relevant metadata is stored in the multimedia database in a different corresponding table for many multimedia categories. Generally, media types have relevant and irrelevant metadata, and each has its unique process to produce metadata. Media types that are related to the domain and are content descriptive are perfect and appropriate support to get the correct interaction. This is because the metadata information about the file should connect all the definitions of the data, and accordingly catch as much media-set details as possible.

C. Pratt reported a technique of retrieving data from a database BLOB (Binary Large Object) data warehouse using SAS as the data analysis tool [32]. The Data Warehouse architecture requires storing summary data in traditional database relational databases and storing raw chip data in a multimedia database BLOB data type. With this BLOB data type many opportunities have opened up for experiment with various methods of retrieving data. Since the databases are fragmented among multiple machines (due to the large data volumes), and to make it easy to register structure required to access the inner parts of the BLOBs, a machine is set aside specifically to direct the client applications and SQL users to the machine their data is on. This machine also provides the information necessary to extract parts of the BLOBs. We refer to this machine as the application director. At the database end, the objects would be too large to be practical. With data volumes in the hundreds of gigabytes, adding descriptive information into the

records would explode the data storage requirements beyond reasonable limits. Objects also allow us to store large numbers of data values.

After the storing of the object, we have to specify how to access this object. This is where the registry comes in. The registry is a set of tables that define the type of object, in this case the type is defined by the application, not necessarily a database data type) and the contents of the object. Each object is comprised of elements that have a name, type, and length. All of this information is stored in the registry. The query looks into the objects and extracts that element, returning it as a column in the user view. An example of a query is as follows:

```
SELECT  
LOT,WAFER,CHIP,GETELEMENT(OBJECT1,D_VAL1)  
FROM DB.TABLE1  
WHERE LOT='123456789' AND  
WAFER='ABCDEF'
```

This query gets the BLOB object1 in the database from TABLE1 table and finds the D\_VAL1 element in each object, returning it as a column in the table.

Y. Velegrakis [33] described that several metadata management tools consider the metadata as an integral part of the data, which means that metadata cannot be retrieved without retrieving also the data with which it is associated. He showed that storing the metadata in independent tables, associated to the data through the q-values, allows them to be queried and retrieved independently. For instance, if a user would like to know the sources that have been used to collect info of a file, he can simply query the metadata table alone.

### **2.3.1 Metadata Classification**

Different kinds of metadata are used to store multimedia information:

- **Content Dependent:** based on the content of the multimedia data. Examples of metadata that rely on the content are the dimensions, colors, and pixels of a file.
- **Content Descriptive:** established on a description of the content of the media data file. This description cannot be extracted automatically.
- **Content Independent:** metadata are independent from the content of the data. Examples of this type are the edit-date of the file, the location taken and hardware type used to record it. There are no details about the file content represented by this type of metadata, yet they are always helpful to retrieve documents from their real physical address.

### **2.3.2 Source of Metadata**

Metadata [10] is selected from different origins that are obtained from system. Three main classes or categories of metadata sources will be described:

- **File content analysis:** the first metadata information origin is the object itself. The media object individually generates the metadata. A content analyzer extracts keywords to fill the multimedia database from media objects; for example, to recognize patterns or shapes inside images and describe them with corresponding words.
- **File context analysis:** Metadata information about the media object can be extracted if the media object is applied in a specific condition and data about that

condition is obtainable. A media object can be found in multiple conditions which help us create many metadata about this single file.

- File usage: Metadata can be also extracted from the environment where the object is used. This type of metadata is more adaptable and active than the other types of metadata sources. The program, which extracts the metadata from the file usage, records and registers the actual use of the file, and obtains the valuable information to be saved.

## **2.4. Retrieval by Shape Similarity**

The shape similarity concept has been universally explored and investigated for many years in the field of multimedia databases. Given an object and a shape to model, the specific program used should evaluate if the current object contains a similar shape as the one described before. The program should take into consideration that the main object could include noise, distortion and deformation compared to the shape in search. Thus, computer scientists have to define the class of possible alteration that an object can experience. For shape similarity retrieval [21], querying in multimedia databases will determine which of the database object is the “same” as the given object.

### **2.4.1 Example**

In a painting database, a user would like to retrieve the Van Gogh’s “Wheat Field under Threatening Skies” (Figure 5 (a)). The user remembers that the bottom part of the image consists of a yellow stripe and that there is a dark blue-sky background. Thus, the user draws a sketch as shown in Figure 5 (b). If, however, the sketch query is given to the system, the search will result with all the paintings in the database, sorted for their

similarity against the sketched image. If the database contains several images similar to the query, the user may need to browse the list in order to find the correct painting.



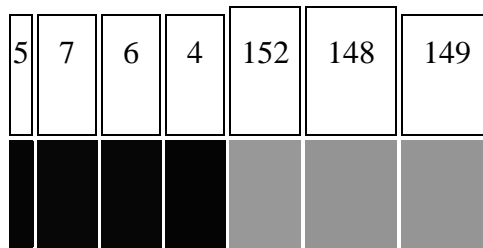
Figure 5: Van Gogh's Wheat Field Under Threatening Skies (a) and the sketch of a query to retrieve it (b)[31]

To analyze images, we should detect the edges and return results about the shapes of the object based on the edge detection method. The edges give information about the shapes contained in the image file. The definition of the edge can be defined as a set of contiguous pixel positions where an abrupt change of intensity of values occurs. Different techniques exist to detect the edges of an image, yet all of them could be categorized into two groups: zero-crossing based and search-based. To detect edges, zero-crossing techniques fetch non-intersections in a second sequence determined based on the image. Before moving to detecting the edges, another stage of flattening or smoothing is usually implemented. Search-based techniques find edges by determining an estimation of edge intensity and then exploring the image for limited directional extreme of angles, working with a determined estimation of the limited edge position.

All the techniques used to detect edges that have been reported vary in the category of applied filters that flatten and smooth the images, in the method that

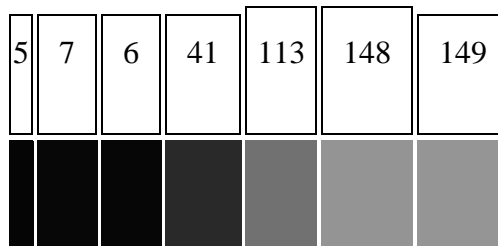
computes the weight of the edge, and in the category of filtering that are used to determine the estimated directions of x and y.

The main complication with edge extrication applying the gradient operators is spotting the edges from only one direction (either vertical or horizontal). To function correctly, the process of image retrieval requires the feature of extrication, which in turn requires extracted edges to relate between returned borders. The detection of edges from an image is not an easy task to be solved. For example, if we want to detect edge from the following signal of single dimension:



We may simply realize that there must be an edge between the 4th and 5th pixels.

If the power of color intensity is much smaller between the 4th and the 5th pixels, and the power of color intensity touching the adjacent pixels were bigger, it could be more difficult to note that an edge exists in this simple area. This case is illustrated in the following signal example:



Therefore, to determine an exact threshold on the power of the intensity variable between two adjacent pixels, the software needs to extract the clear edges, which can be a difficult task to accomplish at times. Actually, this is one reason of many that explain why detecting edges is not a trivial task except if the image object is an uncomplicated representation and the neighboring pixels can be clearly seen.

#### **2.4.2 Edge thinning:**

The edge thinning method is applied to delete undesirable fake spots from image edges. This method is applied following the filtering of the image from any noise using techniques like Gaussian filter, similar to the technique described above. We apply the edge process method to find and extract edges after the edges have been flattened, working with an associated value of threshold. This method deletes all the undesirable spots and produces a single thick pixel edge. The major benefits from the technique are as follows:

1. Intense and fine edges act as a guide for better recognition of objects.
2. Thinning techniques can result in better performance than any other technique applied.
3. Thinning can simply return the parameter of the picture without using complex equations.

There are many popular algorithms used to do this. One such algorithm is described below:

1. Choose a type of connectivity, like 8, 6 or 4.



2. Connectivity 8 is preferred, where all the immediate pixels surrounding a particular pixel are considered.
3. Remove points from north, south, east and west.
4. Do this in multiple passes, i.e. after the north pass, use the same semi processed image in the other passes and so on.
5. Remove a point if:
  - a) The point has no neighbors in the north (if you are in the north pass, and respective directions for other passes).
  - b) The point is not the end of a line.
  - c) The point is isolated.
  - d) Removing the points will not cause it to disconnect from its neighbors in any way.
6. Otherwise, keep the point.

The number of passes across directions should be chosen according to the level of accuracy desired.

### **2.4.3 Multimedia Shape Retrieval Classification**

To retrieve objects from multimedia databases based on their semantic content, end users have to be capable to retrieve media object based on their content by:

- Terms or expressions that involve descriptive texts of the media file or object. For example, end user may want to retrieve a film name by reporting a story line or plot. To answer this query, the system needs to measure similarity of text content and match it with the suitable object.

- Characteristics of the multimedia file. For any multimedia database system, graphical user interface form is generally used to send this sort of queries. An example of this is when the user is able to upload a similar image and requires the retrieval of all images that are close or near the original one. The system should act with the overall image to retrieve the requested file, for example checking the color and shade distribution of the main image. The request can also be sent by the user, who specified a specific color and wants to retrieve all the images that have this same color. To answer these kinds of requested queries, we need a similarity measure of characteristics to extract multimedia objects.
- Visional resources and interconnection structures of the files that appear to the multimedia file. These queries can be sent with query language. For example, a user can upload a picture and request to add some descriptive information to retrieve image from the database based on similarity. Retrieved images represent some interconnected structure with the original uploaded image. Image examination and analysis is required to answer this kind of query to extract complex objects from images.
- Real resources and interconnection of the theoretical objects that appears in multimedia files. Real resources and interconnection of theoretical objects can vary between their visional resources and interconnections in media objects. As an example, the visional resources and interconnections, lower-upper, large-small in a picture can be identical in reality to near-far.

- Time related functions of theoretical objects that exist in multimedia object. As an example, the user can identify many theoretical objects and their time related interconnection, then require retrieving objects that have similar behavior with the original object. To answer this kind of query, an analysis of media objects and specification of the object performance if needed.

#### **2.4.4 Multimedia Shape Retrieval Tool**

To retrieve an object from a multimedia document, we use an Object Extractor tool. This is a semi-automatic tool employed to retrieve objects from media files like videos or images. The retrieved image characteristics are usually the appearance information and the colors used of the objects. The entire image colors can be saved to answer queries that question specific colors. The shape can also be saved to respond to queries that request shape similarity.

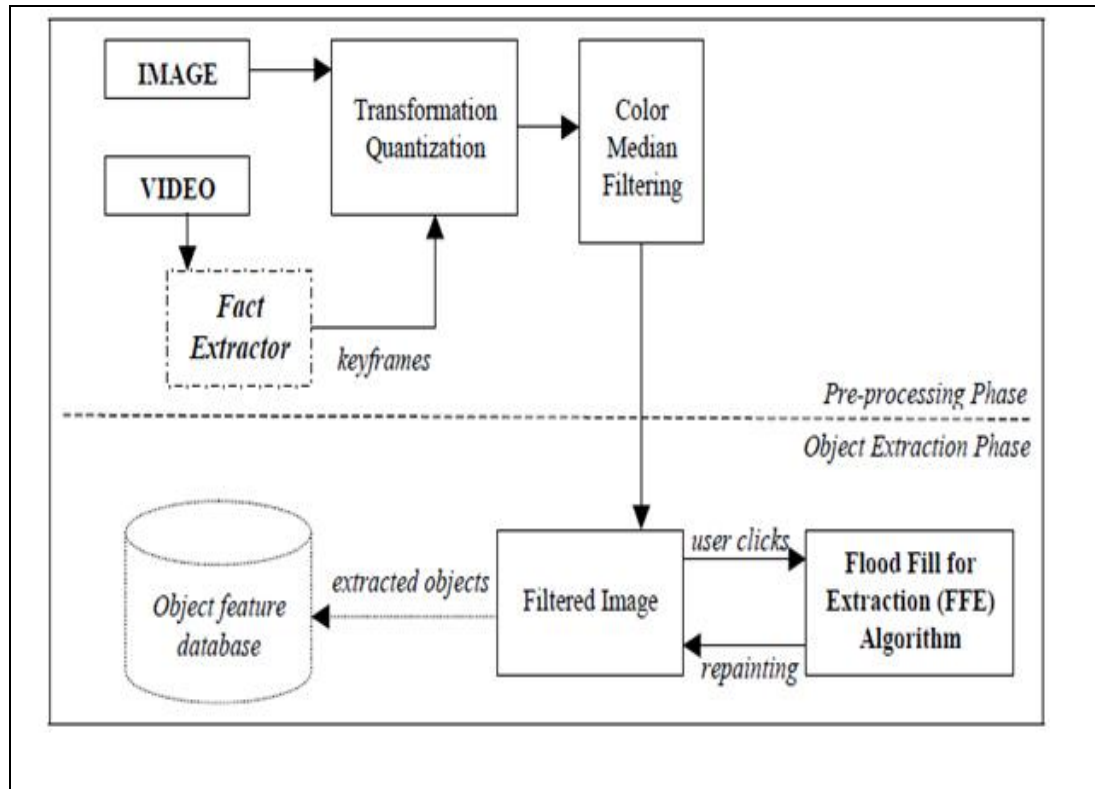


Figure 6:Overall Architecture of Object Extractor.[14]

The Object Extractor tool operates on pictures and videos. The technique applied for the two kinds of data is extremely similar because video slots can be managed as one image. The Fact Extractor tool for video frame database systems manages data of the video and builds key frames of this video. Hence, videos can be handled in the Object Extractor tool via their extracted key frames.

```

procedure FloodFillforExtraction(Pixel p)

// INPUT: a single pixel p

// the INITIATIVE_PIXEL is global to the method and

// it holds the user-clicked pixel

1. if (pixelProcessed(p))

2. return;

3. endif

4. setProcessed(p);

5. if (thresholdPassed(p, INITIATIVE_PIXEL))

6. paint(p);

7. FloodFillforExtraction(left(p));

8. FloodFillforExtraction(right(p));

9. FloodFillforExtraction(up(p));

10. FloodFillforExtraction(down(p));

11. endif

endprocedure.

```

**Figure 7:Flood ll for extraction (FFE) algorithm**

The pseudo-code algorithm given in Figure 7 works with images as an algorithm of Flood Fill for Extraction (FFE). The algorithm works by redrawing a few image pixels and the end user can carry on extracting multiple times according to his/her needs. To start the algorithm, the user has to click on a pixel. This pixel is saved in the “INITIATIVE PIXEL” to be managed later on in the process. The first line examines the end condition, and the fourth to eleventh lines recursively work to continue the process.

We deal with every pixel one time due to the “if statement” that appears in the starting point. For this reason, the algorithm works faster in processing extraction. Line five tests the threshold by assessing the Euclidean distance between color vectors of the two pixels, namely “p” and “INITIATIVE PIXEL.” If this test succeeds, the algorithm redraws the pixel “p” and then recursively calls itself for the neighboring four branches. The algorithm stops working when no branches left to be executed in the recursive tree.

## **2.5. Content Base Retrieval**

Content-based retrievals are more desirable in multimedia database systems [15]. For example, searching for a suspect according to a witness's description in a large image database of criminal faces is a very arduous task to complete. In this case, content-based retrieval is the most preferable, since it has the nature of visual and similarity-based methods. Sometimes, content-based retrieval can become fuzzy because the result is not always exact. For example, there may be several hundred thousand facial images in this criminal identification system. “Finding something similar to another object” explains the similarity-based query types of content based processing [11]. For images, we can test the color ratio, patterns, shapes and relations between objects in an image. For sounds, we can test a melody or a note model inside a section of audio. Also, spoken words in a song are likely to be recognized.

To retrieve multimedia files using the content-based technique, visual methods are dropped and alternate search based method of the content of the multimedia file itself is applied. Numerous studies have explored the techniques of retrieval based on the content of the object. For images and video shot, content can cover the shades, colors, material, shape, etc. For audio files, the content can include notes, melody, rhythm, etc.

For retrieval by content, the media files are saved as computerized representation of the media objects. To retrieve information from a media file, the results are always fuzzy. The user will not get the exact requested results he/she desired. To help in retrieving the precise file, indexing and metrics should be used, but the user will always have the last decision in recognizing a query results. The query below fetches all images that contain a person:

```
SELECT m  
FROM Images m, Persons p  
WHERE m contains p
```

Two types of errors can be found when running this query:

1. False results returned by the query as a solution, and
2. Absent results that need to be a result, but they are not found in the query answer.

To compute the success of any query in content-based retrieval, we use two metrics that are described as follows:

1. First the precise answer, which computes the correspondence between the number of objects returned and the total returned object number by the query as a response.
2. Second the recall, which computes the correspondence between the accepted returned objects and the entire number of accepted objects in the whole group.

The two measures above can vary between [0, 1] as an answer. The purpose is that they return a value closer to 1 in order to be adopted. The elements that guide them both hang on the application. Both measures only have conceptual values because the end used is the only person who will declare the degree of correctness of any results returned by the query. Normally, any image content is composed of many objects that form this image. The application decides if those objects inside the image are valuable to the query posed. These vary on many levels considering the location where this object is placed inside the image and other descriptive properties.

### **2.5.1 Content Base Retrieval for Images**

The need to discover and find a specific image from a collection of different kinds of images is becoming a necessity to many academic fields and professions including design engineers, journalists and investigators, medical professionals, trend setters, clothes designers, planning professionals, construction workers, and crime prevention specialists to name a few. Not much has been brought public for users to search images. Attempts are now being worked on to classify the user's way of behaving to permit for better results in the future.

To search an image saved in multimedia database, the image is split into equal sized rectangular cells that are labeled segments. The process of segmentation will be described later on. A connex region  $\mathfrak{R}$  is a cell set, such that if  $(x_1, y_1) \in \mathfrak{R}$  and  $(x_2, y_2) \in \mathfrak{R}$ , then a cell sequence exists:  $C_1, C_2, \dots, C_n$  in  $\mathfrak{R}$  and  $C_1 = (x_1, y_1)$ ,  $C_n = (x_2, y_2)$  and Euclidian distance between  $C_i$  and  $C_{i+1}$  is 1, for  $i$  ranging between 1 and  $n$ . A homogeneity predicate that associates with an image is a function  $H$  that gets a connex region  $\mathfrak{R}$  from the image and returns true or false (for example,  $H$  is true is more than



100x $\delta$ % from the cells in the respective region have the same color -  $\delta \in [0,1]$ ). An image segmentation according with the predicate  $H$  is defined as being a set of regions  $R_1$ ,

...,  $R_k$ , so that:

$R_i \cap R_j = 0$  for any  $1 \leq i \neq j \leq k$  and  $I = R_1 \cup \dots \cup R_k$

$H(R_i) = \text{true}$ , for any  $1 \leq i \leq k$ , and

for any  $1 \leq i \neq j \leq k$ , if  $R_i \cup R_j$  is a connex region,  $H(R_i \cup R_j) = \text{false}$ .

Generally, the method does not take into consideration all image pixels since the pixel number can be very high. A typical approach is to modify the matrix of the image in a compromised description. The methods that are mostly used are Discrete Fourier Transform (DFT), Discrete Cosine Transform (DCT) and Wavelet. Two procedures are available for content-based retrieval linking two images: function based on distance and function based on the transformation cost. Databases currently represent images in two ways: either as relationships or as dimensional data structures. Generalized R trees are similar with R trees, except that they contain a set of Generalized Bounding Rectangles (GBR), which are represented by  $2 \times (n+2)$  fields that correspond to the lower/upper bounds for each dimension.

### **2.5.2 Content Base Retrieval for Video**

A video is a chain of scenes that are collected by snaps, which are frame series. The frame can be described as a fixed image. A snap (or shot) is a continuous act related to time and space. Finally, a scene is a series of snaps with similar semiology [6, 26]. To work on a video, the video is required to be logically divided into uniformed parts of segments. This function of segmentation is the earliest step in order to start the content-based search on video, facilitating the search of specific objects in a video. The methods

of video segmentation find out where the snaps have been chained or linked together [9]. First, to analyze and browse the content of a video, it is divided into snaps (or shots). As described above, a snap is a series of image that represent a continuous act from one camera, which captures many snaps in milliseconds. Then, those snaps are linked together to build the final and complete video. Snaps may be described as the smallest part of a video. We need to analyze these snaps to better organize the content-based retrieval through inter and intra snaps connections. Video segmentation segments the initial image frame and then it marks the development of the objects in motion. After the segmentation process of image objects for every frame, many implementations can be done on these segmented objects once they have been extracted, like studying the object in each segment and content-based retrieval of a video. A video is described to be a set of snaps connected together using specific tool [5]. To extract objects from video segments, the detection of time related border (as shown in figure 8) is required.

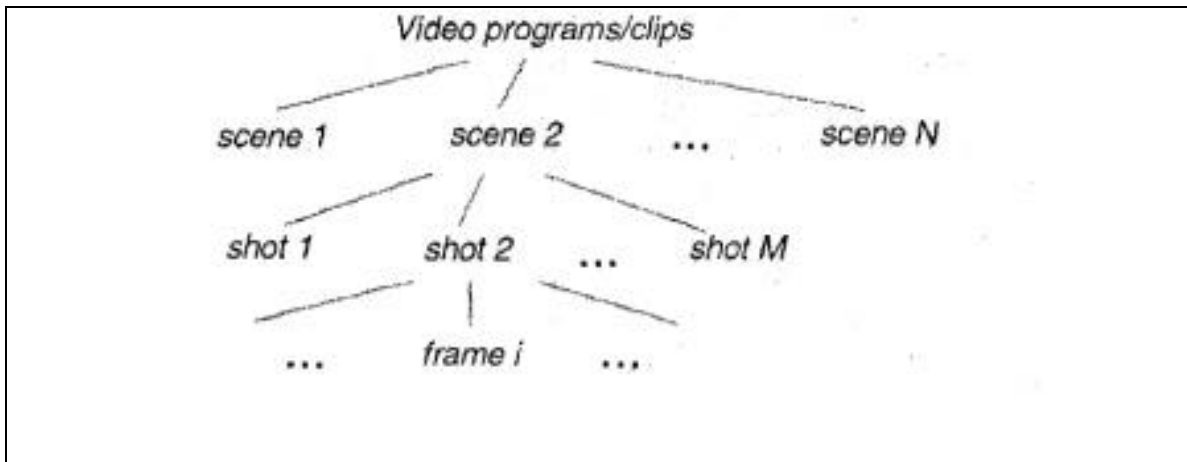


Figure 8. Content based video retrieval systems.[5]

The next step is to represent compressed segments as FS (Frame-Segment) or RS trees. The FS trees propose to mainly create a table to be associated with every object found in

a segment. The FS Tree joins each node to the area in which the frames from the sub-tree that is dominated by that node are found. The indexing method associates each range with two tables, one with objects and one with activities. Each of the elements of these tables is an ordered list of pointers to the nodes from the FS tree, which contains the respective objects/activities. The RS trees are similar to FS trees with one major exception. The concepts of object vector and activity vector remain the same, but the (start, end) frames, which are actually rectangles of e-s length and width zero, are stored instead in an R tree. That will be extended to show which are the objects/activities for each rectangle. The RS tree's advantage is that on each disk access they bring into memory more than one rectangle, those being in fact proximate rectangles. The main purpose of video segmentation is to recognize the file partition in a recorded video. Every file has to be sufficiently small to maintain only one subject, at the same time it should be sufficiently long to permit the system to decide if it is applicable or not. Yet, the video can have other specifications that could help in finding the file partition. For example, the detection of speech/non speech can identify a change in the subject.

The system will compare the color and orientation histograms in adjacent frames, analyze motion flow and track audio. The main problems in segmentation systems are transitions like wipe, fade, and cross-dissolve; camera motions such as pan, zoom, etc.; and moving objects occupying a large percentage of the image.

### **2.5.3 Content Base Retrieval for Audio**

Research and analysis of the retrieval by content-based domain has mainly focused on retrieving image video data. But, with the rise of services like Voice over IP and fast packet switching networks, audio-content based is now a highly attractive study

area. Many recorded audio files deal with multimedia implementation. They are deployed effectively if the system has the capability to categorize and retrieve recorded voice objects based on sound effects. Fast growing audio database needs require the search of a technique that could efficiently search for audio files based on content. [13, 17, 23, 24].

Similar to image and video content, any audio content could be drawn out by special characteristics like amplitude, frequency, etc. The most used methods for audio-content are to divide the signal based on time to have smaller parts that contains similar properties. The division is done with a single step utilizing similarity predictor. As soon as the audio is divided into segments, it can be seen as smaller slots series  $w_1, w_2, \dots, w_n$ . Every slot has “k” relevant features that can be extracted. Thus, we get “n” points in a  $(k+3)$ -dimensional space (audio source file, the window and its duration add to the k features of the signal). Obviously, this approach is unrealistic since only ten minutes can produce 100.000 windows. Therefore, adapted compression techniques (DFT, DCT) have to be applied in this case, too. If the user requests a query as: select each audio that contains a close sound, a DFT will work with this query to search the nearest results.

In a major new research, a system named “Muscle Fish” is introduced. This project differed from other content-based retrieval of audio by its ability to efficiently retrieve objects. Many properties are considered in this “Muscle Fish” system like bandwidth, pitch, tone, accent, and vibration. They used these features to present an audio file. Rules used to classify the audio in groups inside the media database are a normalized Euclidean distance and the nearest neighbor (NN).

In addition to the “Muscle Fish” system, a search done by Liu et al. [6], the same properties are employed, with the addition of sub-band energy ratios. To separate different groups, sounds are evaluated using the intra- and inter-class feature to specify the extremely harmonized property, then a grouping of the objects is done by the use of a neural network.

# Chapter Three

## Using Compressed Bit-Vector for Multimedia Data Retrieval

### 3.1 Introduction

This chapter is dedicated to the description of a new algorithm using a compressed bit vector for multimedia data retrieval, which will help in accelerating the query response time in multimedia databases using any retrieval type. The multimedia system evolution that successfully experience retrieving and giving important information extracted from a huge multimedia database system mostly rely on the proper and existing implementation of the media accessing methods corresponding this application. The existence of an enormous volume of media data files questions the aspects of the management of multimedia objects and the problem of implementation. Typically, queries in multimedia database are multidimensional and have complex selections. Users that request specific queries in multimedia databases usually find it hard to find answers to all requirements. Due to these characteristics, Bit-vector indexing techniques have shown promising results for processing multimedia databases. A significant advantage of the bit-vector technique is that complex logical selection can be performed very quickly via bit-wise AND, OR and NOT operators. In this paper, we further explore the issues of query acceleration using bit-vectors, and we concentrate on

optimizing one of the query operations “Selection,” which is further discussed with simple queries, and the more complex queries using the four different types of joins: hash join, inner join, merge join and nested loop join. Although bit-vectors can be space inefficient for high cardinality attributes, but the space for the compressed bit-vectors works best compared to other techniques.

A bit-vector is a vector or array of data that stocks bits briefly. A bit vector is time composed from the bit values of the collection  $\{0, 1\}$ . Bit-vector is a term applied here to denote a large classification & indexing plan that stocks index as bit sequence. A bit-vector is a bit string in which each bit is mapped to a record ID. A bit in a bit-vector is set to 1 if the corresponding ID has a property “P” and is reset to 0 otherwise. The property “P” is true for a record if it has the value “x” and attribute “X.” The query selection can also involve many attributes. Many bit-vectors have proved to work efficiently in database implementation. Bit-vectors permit vectors of bits to be stocked and handled in the memory set for extended time phases. Bit-vectors can potentially explore bit-level similarity, utilize the data cache to the max, and minimize access to memory. Bit-vectors usually work best in different data forms on reasonable data sets, and on those that are efficient asymptotically. To further improve their effectiveness, we will study their compression scheme, which will potentially minimize the area used without expanding the managing time of the query.

Generally, a bit-vector is stocked as a group of bits and the majority of operations on regular bit-vectors are logical bitwise operations. Considering our concerns in using the bit index on huge databases, the main aid is to reduce the sizes of the index. Plus, we also wish to be able to efficiently execute logical operations on the compressed bit-

vectors. A problem with using uncompressed bit-vectors is their large size and possibly of high expression assessment costs when the indexed attribute has a high cardinality. A single technique to deal with using bit-vectors on high-cardinality attributes problem is to store them in a compressed bit-vector form. Using compressed bit-vectors has multiple advantages that potentially adjust performance: minimized disk space needed to stock the indices, faster reading of the indices from the disk into the memory, and more cached indices in the memory with this compressed form [19]. Some Boolean operation evaluation algorithms, which operate on compressed bitmaps without having to decompress them, might be faster than same operations on the regular bit-vectors. The scheme for compressing data, in addition to transforming data, guides the reducing of enormous volume required. The technique here is to alter the issued multimedia data bit-vector to another modified area to eliminate the redundancies in the real data.

A bit vector “B” of “u” bits can be represented as  $B[0::u)$ . It can be stored in  $uH1(B)$  bits so that the operations can be answered in constant time. We will only save the 1-bits in if the response to the query is true. With this representation of “B,” we can access any block of size “b” in constant time, which is sufficient for implementing rank and selecting as we just saw. In addition, access queries can be answered in constant time, too.

Decompression is made from the backwards process to re-transform and decode the data to its native origin form. This operation generally encounters some data loss, which is a major problem of multimedia applications. Our algorithm will try to ensure negligible loss of data when retrieving information.



### 3.2 Algorithm Details

Our algorithm, compressed bit-vector for multimedia data retrieval, uses bit vectors to return exact answers to any query in multimedia databases, with any retrieval process used. For example, a specific shape may be compared to a number of pictures in a multimedia database to find a picture or many pictures with the same characteristics. The search may result in either one or more matches found, or no matches at all in a set of objects in the multimedia database.

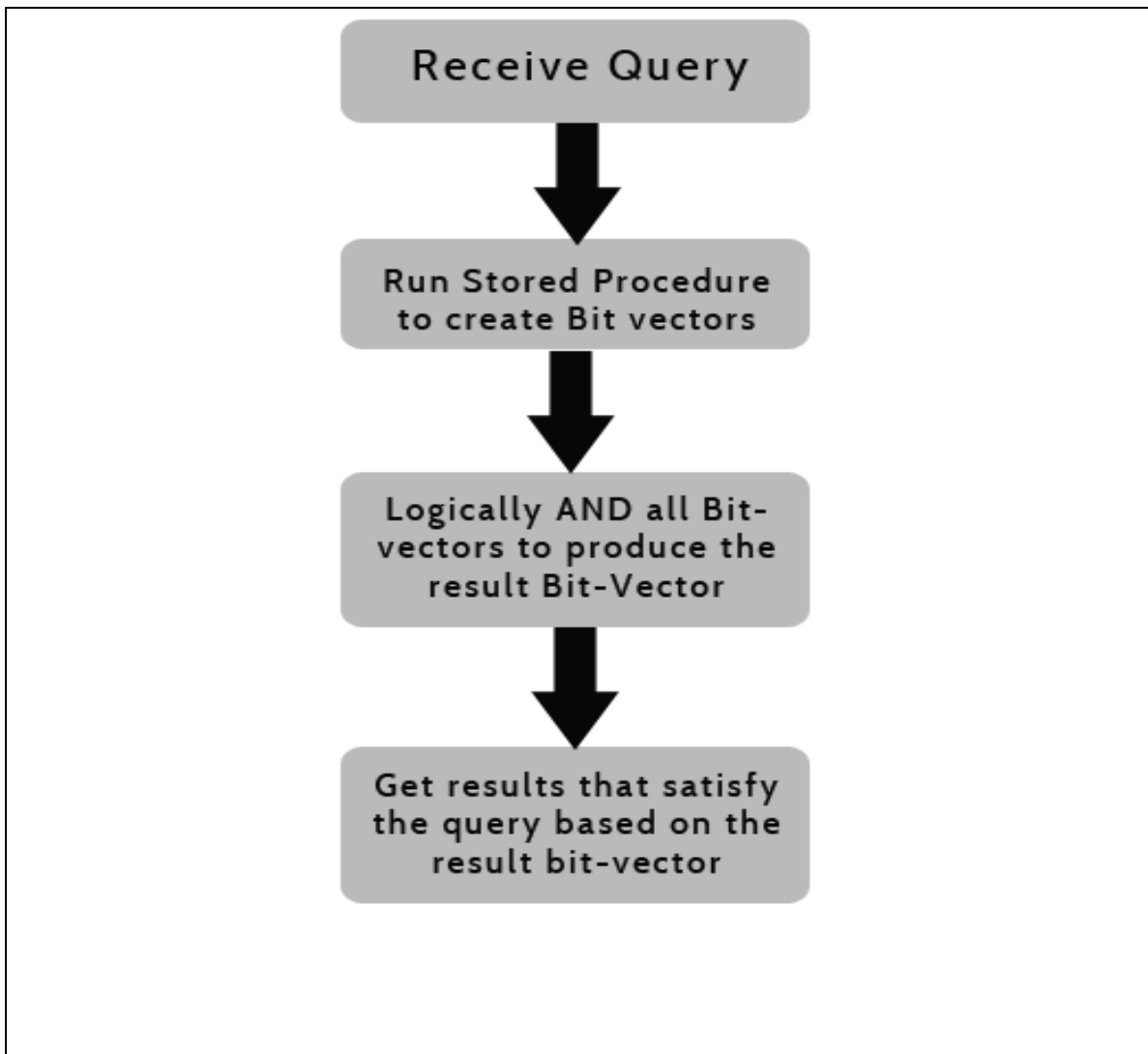


Figure 9: Algorithm work flow

The above Figure 9 is an exemplary operation on how a query can be handled in searching for a specific attribute in a multimedia database. First, a receive query operation receives a query item. When a user requests a query in multimedia database with some attribute, a bit vector index is created for each attribute. Each bit vector index indicates whether each of the attributes in the selected database does or does not exist in any of the retrieval strategies used. When a query is received, the bit vector indices associated with each of the selected attribute values are then logically ANDed together to form a single result bit vector index. The result bit vector index identifies a reduced set of accepted IDs of the data table containing the multimedia objects. This reduced set of IDs in the multimedia data objects returned by the bit operations may then be quickly searched using a linear scan to determine a match or matches for the query point. To retrieve resulting matches, we simply select the IDs of the query table that contain a “1” bit in the bit-vector.

Following is the stored procedure used in building the bit vector of the specified attributes for any query in multimedia database. For simplicity and straightforwardness, we used the “retrieval by meta and logical attributes” strategy in a real university database.

```

DELIMITER //

DROP PROCEDURE IF EXISTS mysql_BitVectorTable //

CREATE PROCEDURE mysql_BitVectorTable ( IN attributeValue VARCHAR(255))
BEGIN

    DECLARE idSelected  VARCHAR(255);

    DECLARE exit_loop BOOLEAN;

    -- Cursor for select statement

    DECLARE query_cursor CURSOR FOR SELECT id FROM students where
city = attributeValue;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET exit_loop = TRUE;

    DROP TABLE IF EXISTS bitvector;

    -- create a new table in database with id and Boolean

CREATE TABLE bitvector (id VARCHAR(7),bitValue BOOLEAN);

    OPEN query_cursor;

    query_loop: LOOP

        FETCH query_cursor INTO idSelected;

        -- save in bitvector

        INSERT INTO bitvector (id,bitValue) VALUES (idSelected,1);

        IF exit_loop THEN

            CLOSE query_cursor;

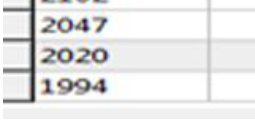
            LEAVE query_loop;

        END IF;

```

```
END LOOP query_loop;  
  
END //  
  
DELIMITER ;
```

After the construction of the bit vector, it will be stored in the database as a regular table. Each bit vector contains two fields: The first corresponds to the original table index, and the second contains the bit 0 or 1 referring to the absence or presence of the main query attribute. Each bit vector should contain the same number of indexes as the original table. But to compress our bit vector, we will only save the 1 bits associated with the presence of the query attribute and remove the 0 bits from the bit vector. Thus, the bit vector will contain a smaller number of bits and minimize the response time of the process. Figure 10 is an example Bit vector table resulting from the previous stored procedure call. This bit vector contains only the 1 bits as shown in the compressed bit vector with the index of the original table.



2047	
2020	
1994	

Figure 10: Bit vector table

### 3.3 Experimental results:

To explain our algorithm, we will interpret and evaluate the finding results. The following discussion will focus on the application, appropriateness and usefulness of the bit vector algorithm for multimedia database retrieval using metadata attributes that represent the simplest way to retrieve media files.

The first research query was to select all information about students that belong to a specific campus in a specific major. We ran our algorithm on a database table containing multimedia files. We used a traditional database application that uses fixed sized data, but the multimedia size of data can vary dynamically. All unformatted data (mainly text and images) has been handled in this database system through BLOBs. They

usually support only a few generic operations, such as reading or writing parts of BLOB. The first table used is the student application table with student images in each record. The table includes more than 51,000 records of student information. The tested query involves retrieving the student images that match certain required parameters. The outcome result will determine the time it took to handle this simple query.

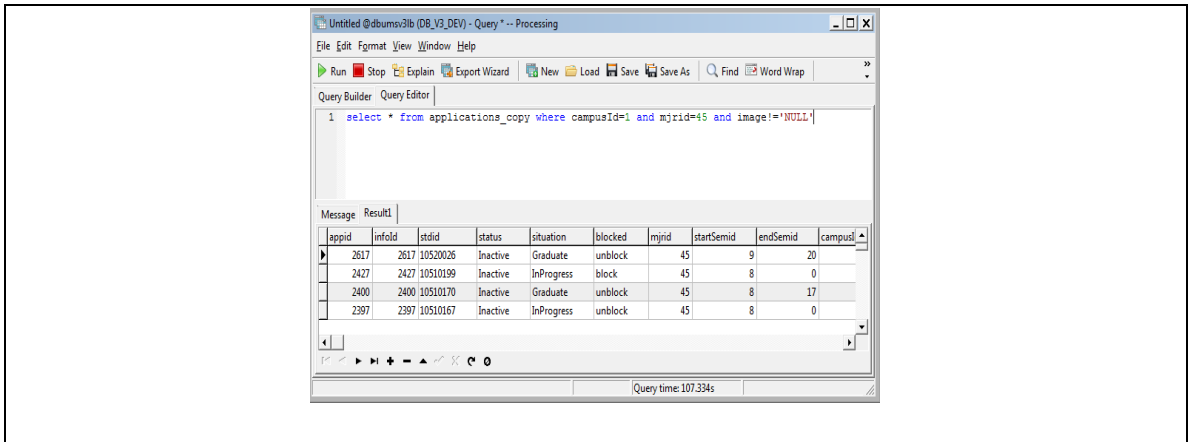


Figure 11 :Query Example with processing time

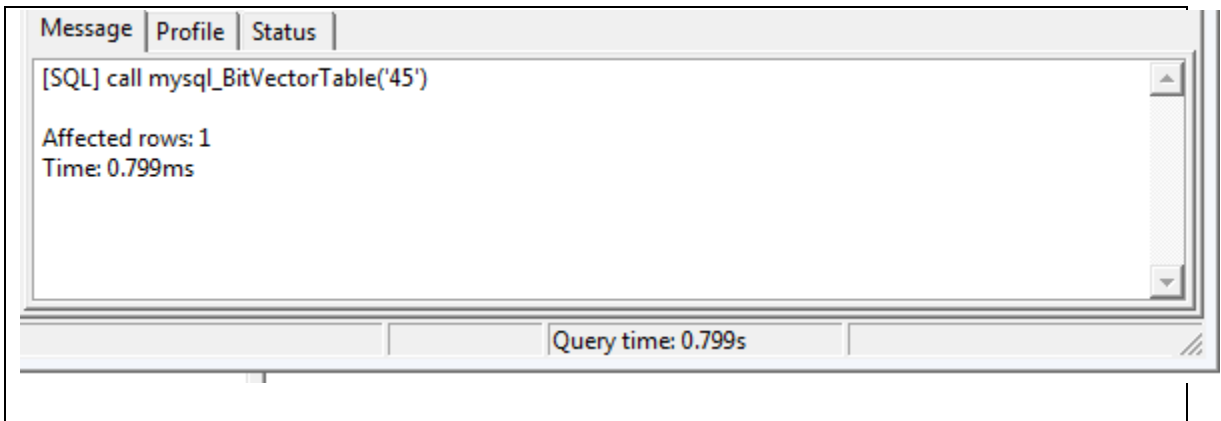
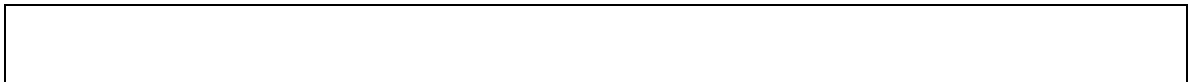


Figure 12 :Call of the stored procedure



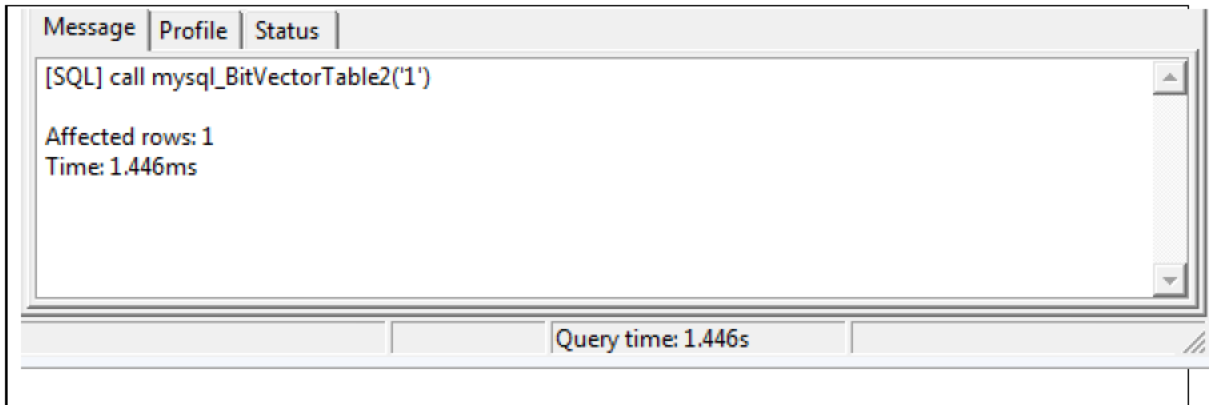


Figure 13: Second call of stored procedure

```
//Call the stored procedure to create the first bit vector for the first attribute

$sql="call mysql_BitVectorTable('45')";

$webUserDB->Execute($sql);

//Call the stored procedure to create the second bit vector for the second attribute

$sql2="call mysql_BitVectorTable2('1')";

$webUserDB->Execute($sql2);

$query="select id from bitvector";

$getQuery=$webUserDB->Load($query);

// get the first bitvector

foreach($getQuery as $k=>$v){
```

```

        $arrayBitvector[]=$v['id'];
    }

    $query2="select id from bitvector2";

    $getQuery2=$webUserDB->Load($query2);

    // get the second bitvector

    foreach($getQuery2 as $k2=>$v2){

        $arrayBitvector2[]=$v2['id'];

    }

    // AND in the two bitvectors according to the original query

    foreach($arrayBitvector as $kFinal=>$vFinal){

        if(in_array($vFinal,$arrayBitvector2)){

            $arrayFinal[]=$vFinal;

        }

    }
}

```

Figure 14: PHP process returning the final result



In this simple query, the program indicates that it requires an execution time of 107.334 seconds. This means that there is a need for a method to run queries and return results in a more efficient time. The stored procedure, described above, is used to build the bit vector for the same simple query. A stored procedure is built for every attribute value in the query. After selecting the first attribute, a bit vector table is created and saved in the database. A second bit vector is created for the second attribute. Figures 12 and 13 show the time to create both bit vector tables. Creating both bit vector took:

$$0.799+1.446 = 2.245 \text{ seconds.}$$

Next, we will “AND” all bit vectors created to maintain the final bit vector. Figure 14 illustrates how both bit vector tables are “ANDed” using a PHP function. Using a time calculator, the retrieval of student images took 3.84 seconds to display on the website.

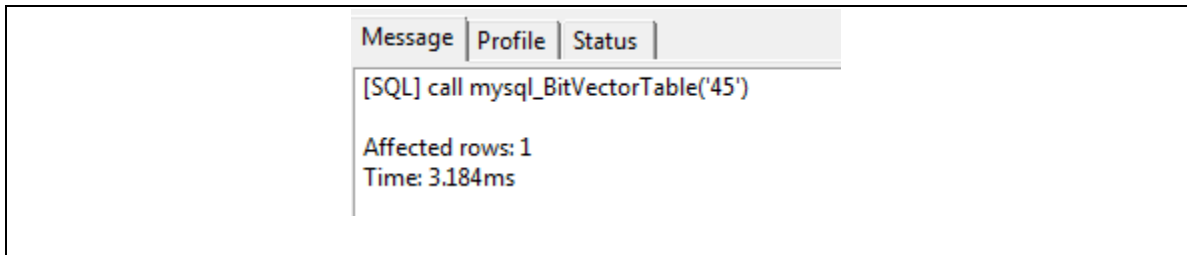
We have also tested our algorithm on different kinds of queries. Other than the simple query noted above, we used two attributes for tables with an index. We ran our algorithm on simple queries using two attributes for tables without index, then for complex query using hash join, inner join, and nested loop join. Results obtained will be saved and discussed. To test our algorithm on another more complex query, we will use the “inner join” type. For example, we will try running our algorithm with the following query:

```
SELECT id  
  
FROM applications  
  
INNER JOIN majors on applications.mjrid=majors.mjrid
```

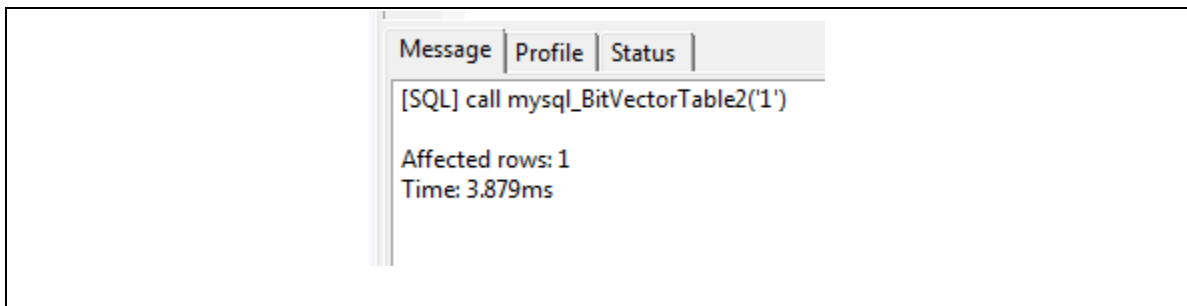
**WHERE attribute1='a' and attribute2='b'**

The time it took to build the results of this query in the regular case is: 112.182 seconds.

For our algorithm, the time registered to build both bit vectors is as follow:



**Figure 15: Bit vector building time**



**Figure 16: Second Bit vector building time**

Furthermore, the processing time to display the result is: 3.6691 seconds.

The required total time for the overall process is: 10.73 seconds

The previous results show the efficiency and rapidity of searching of multimedia data using the bit vector algorithm with the metadata retrieval system. The following table shows the time of different kinds of queries with and without applying our algorithm:

**Table 2: Running time for different Queries**

Query Type	Running Time Without Bit-Vector Algorithm	Running time With Bit-Vector Algorithm Using Stored Procedure
Query with Attributes For Table With Index	107.33 seconds	6.87 seconds
Query with Attributes For Table Without Index	121.54 seconds	11.28 seconds
Query with Inner Join	112.18 seconds	10.73 seconds
Query with Hash Join	106.53 seconds	5.53 seconds
Query with Nested Loop Join	107.87 seconds	6.71 seconds
Query with Merge Join	107.12 seconds	6.54 seconds

To further develop our algorithm, we wrote it without a stored procedure function. A Code that generates the bit vectors stored on the web server functioned as the bit vector. The query selected each attribute alone to retrieve the IDs that match the query results. Then the bit vector was saved in the memory using a key and a value. The key corresponds to the media file ID in the database, and the value corresponds to {0, 1} of the bit vector. To compress our bit vector, we only saved the 1 bits in the memory. After saving the bit vectors for each attribute, we added the “AND” or “OR” in the bit vectors according to the query requirements to get the final IDs that respond to the query result.

```

<?php

//The query require the retrieval of all students pictures that are in Beirut campus and
have the major set as Computer science

//All The attributes required for retrieval are set in a PHP array

$arrayOfattribute=array("campusId"=>'1',"mjrid"=>'45');

$tableName="applications_copy";

//Foreach Attribute get the selection

foreach($arrayOfattribute as $key=>$value){

    $query="select appid from ".$tableName." where ".$key." = '".$value.'";

    $getQuery=$webUserDB->Load($query);

    $arrayBitVector[]=createBitVector($getQuery);

}

foreach($arrayBitVector[0] as $kFinal=>$vFinal){

    if(in_array($vFinal,$arrayBitVector[1])){

        $arrayFinal[]=$vFinal;

    }
}

```

```

}

//ANDing

//$finalArrayOfIds=ANDing($arrayBitVector);

//ORing

//$finalArrayOfIds=ORing($arrayBitVector);

print_R($arrayFinal);

//function to create the bit vector

function createBitVector($tableToUse){

    foreach($tableToUse as $k=>$v){

        $bitArray[$v['appid']]=$v['appid'];

    }

    return $bitArray;

}

//function to Oring all the associative arrays

function ORing($arrayToUse){

    foreach($arrayToUse as $keyBitVector=>$valueBitVector){

```

```

        foreach($valueBitVector as $k=>$v){

            $finalArray[$k]=$v;

        }

    }

    return $finalArray;

}

//function to ANDing all the associative arrays

function ANDing($arrayBitVectorToUse){

    $finalArrayOfIds=array();

    foreach($arrayBitVectorToUse as $keyBitVector=>$valueBitVector){

        if(empty($finalArrayOfIds)) $array1=$valueBitVector;

        else $array1=$finalArrayOfIds;

        if(isset($arrayBitVectorToUse[$keyBitVector+1])){

            unset($arrayPreOfIds);

            foreach($array1 as $k=>$v){

                if(in_array($v,$arrayBitVectorToUse[$keyBitVector+1])){

```

```
        $arrayPreOfIds[$k]=$v;

    }

}

if(isset($arrayBitVectorToUse[$keyBitVector+1]))

unset($finalArrayOfIds);

$finalArrayOfIds=$arrayPreOfIds;

}

else{

    if(count($arrayBitVectorToUse)==1)

$finalArrayOfIds=$valueBitVector;

    else{

        $arrayPreOfIds=$finalArrayOfIds;

        unset($finalArrayOfIds);

        foreach($array1 as $k=>$v){

            if(in_array($v,$arrayPreOfIds)){

                $finalArrayOfIds[$k]=$v;

            }

        }

    }

}

}
```

```

    }
}
}
}
return $finalArrayOfIds;
}
}
?>

```

Figure 17: PHP Algorithm

We ran and tested the algorithm that uses PHP to build bit vectors on the same query types tested before. Results are presented in Table 3.

Table 3: Second Running time for different Queries

Query Type	Running Time Without Bit-Vector Algorithm	Running time With Bit-Vector Algorithm Using Stored Procedure	Running time With Bit-Vector Algorithm Using PHP Code
Query with Attributes For	107.33 seconds	6.87 seconds	7.64 seconds



<b>Table With Index</b>			
<b>Query with Attributes For Table Without Index</b>	121.54 seconds	11.28 seconds	12.33 seconds
<b>Query with Inner Join</b>	112.18 seconds	10.73 seconds	12.81 seconds
<b>Query with Hash Join</b>	106.53 seconds	5.53 seconds	7.55 seconds
<b>Query with Nested Loop Join</b>	107.87 seconds	6.71 seconds	8.12 seconds
<b>Query with Merge Join</b>	107.12 seconds	6.54 seconds	7.22 seconds

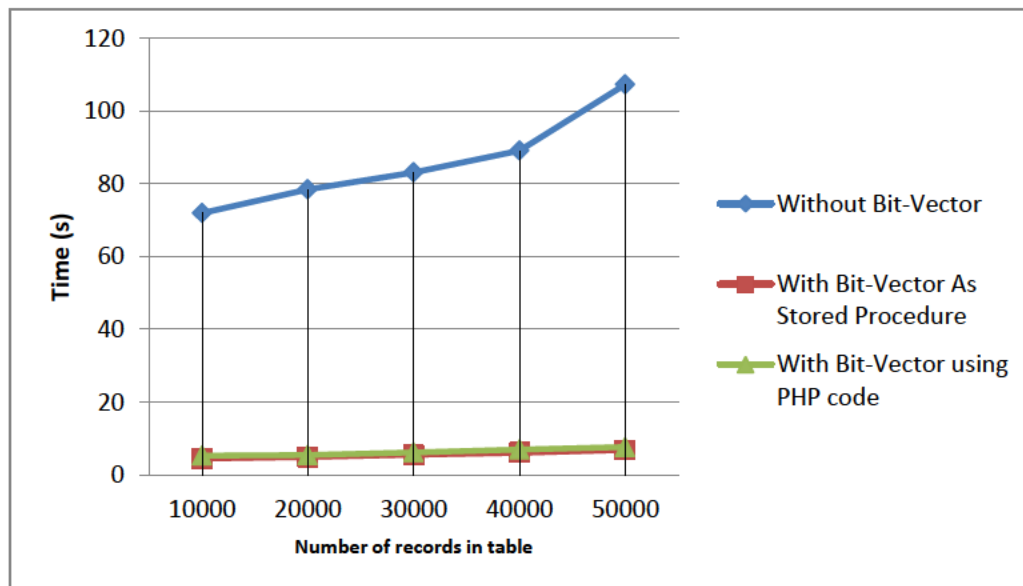


Figure 18: Performance analysis for a simple query in multimedia database

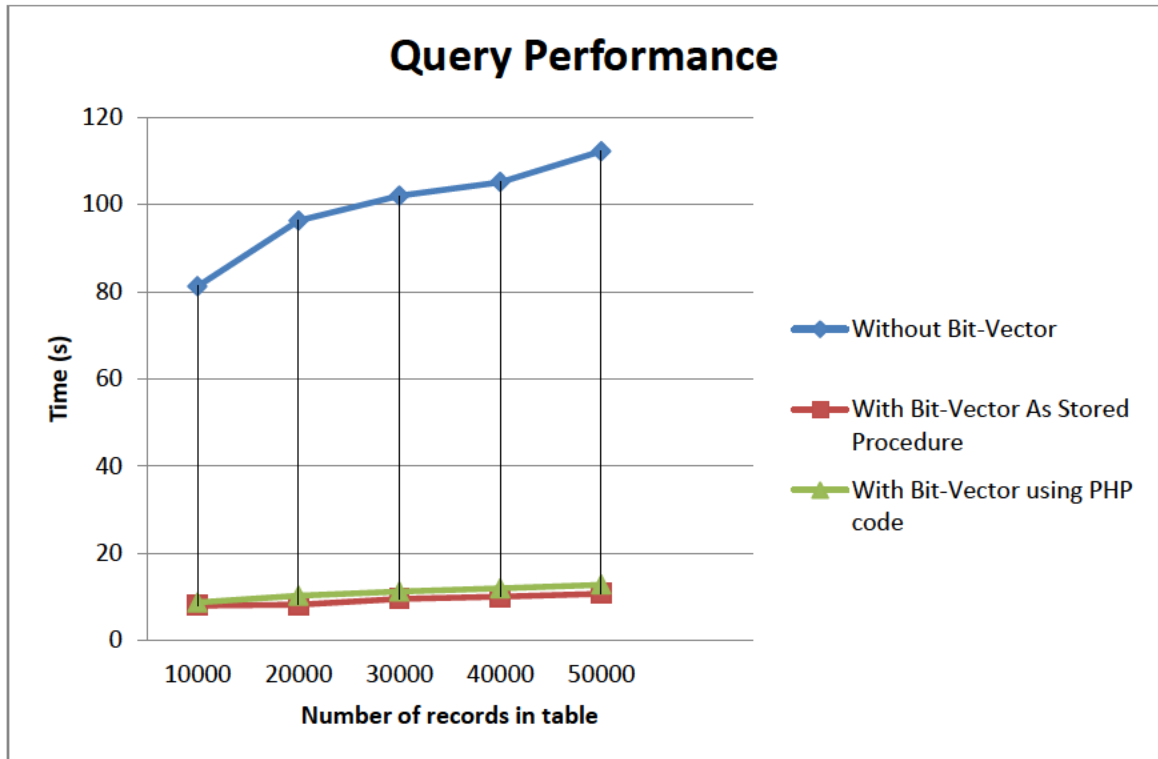


Figure 19: Performance analysis for inner join query in multimedia database

Figure 18 and 19 show the performance of the two algorithms compared to the running time of selection process without applying any algorithm. To calculate the complexity of our algorithm, we defined time taken by the algorithm without depending on the implementation details as our algorithm runs in linear time. The execution time is exactly proportional to the size of input. The algorithm complexity is of order  $O(n)$ .

### 3.4 Conclusion:

A new strategy has been proposed for retrieving multimedia data objects stored in a database. We searched for specific queries selecting objects from a multimedia database such as searching for particular images stored in the database. As a result of the search, either one or more true results are found, or no result exists in the set of objects in the database.

Our bit vector for retrieving media files algorithm was proposed and tested on real data. In fact, Bit vector indexing techniques have shown promising results for processing multimedia databases. We have explored the issues of query acceleration using bit vectors, and we have concentrated on optimizing “Selection” in query operations, which, applies with simple and more complex queries using the four different types of joins: hash join, inner join, merge join and nested loop join. To optimize the results returned, our method uses a compressed bit vector to save the accepted rows of information. This method guarantees fast and efficient query results. This technique also minimizes the cost and amount of data transferred. Our test results show that the simplest approach towards solving queries in multimedia database is the linear scan. This approach outperformed more complicated approaches.

# Chapter Four

## Conclusion

### 4.1 Conclusion

In any system, a query is produced, assembled and compiled. Then, an implementation tree is generated to obtain results. Optimization is more complex considering the heavy, confused or fuzzy expression or retrieval by content-based. Furthermore, optimization has to be globally approached. To generate output, the system considers that query expression weights and value of attributes have been employed to compute the importance of several outputs and to present them to users. The output results need to be registered quickly and have better quality.

Almost all multimedia data files have to be categorized as an n-dimensional. These data files require particular indexing and retrieving techniques. The urge to question the performance in the retrieval action require work with multi-dimensional indices, media object clustering and content-based retrieval. As these kinds of querying are usually slow in multimedia databases, a new customization technique to retrieve multimedia files was needed.

In spite of the remarkable advancements of conceptual research in retrieving multimedia data, there has been small influence on the speed of any query in multimedia database retrieval. One promising approach is to use a compress bit vector for multimedia

data retrieval to fast select and or combine appropriate features. The method described above facilitates rapid searching of multimedia data objects in a multimedia database. When a query is received, each attribute in the multimedia database is divided into a number of bit vectors. A single bit vector is then formed, returning the joint different results of each attribute bit vector. This bit vector is then used to determine matches for the main query, returning a reduced set of multimedia objects instead of the entire multimedia data object, thereby greatly reducing the query search time and increasing the efficiency of the process by allowing the mix of integer and bit-level operations. The compressed bit vector method is used to perform operations quickly, to reduce the query response time and to minimize the cost and amount of data transferred.

## **4.2 Future Work**

We are currently working on using this compressed bit vector to construct abstractions to be used for more powerful concurrent query analyses in multimedia databases, such as saving repeated queries in existing libraries. This may lead to more efficient and faster query response time.

# References

- [1] A. Analyti and S. Christodoulakis, “Multimedia Object Modelling And Content-Based Querying”, Multimedia Systems Institute of Crete (MUSIC), Crete Technical Univ., Dept. of Comp. Sci., Greece, Technical Report Chania 73100, 2000
- [2] A. Burad, “Multimedia Databases”, Indian Institute of Technology, Dept. Comp. Sci., India, 2006
- [3] G. Chechik *et al.*, “Large-Scale Content-Based Audio Retrieval from Text Queries”, *MIR '08*, Vancouver, British Columbia, Canada, October 30–31, 2008
- [4] K. Cox, “Information retrieval by browsing”, Hong Kong Univ., Dept. of Comp. Sci., Technical Report, 1992
- [5] R. Fagin, “Fuzzy Queries in Multimedia Database Systems”, *IBM Almaden Res. Center*, California 95120-6099, 2000
- [6] J.T. Foote, “Content-Based Retrieval of Music and Audio”, Ph.D. dissertation, National Univ. of Singapore, Institute of System Science, Singapore 119597, 1999
- [7] D.A. Forsyth, “Benchmarks for storage and retrieval in multimedia databases”, , Ph.D. dissertation, Berkeley Univ., Dept. of Comp. Sci., CA94720, 2002
- [8] D. Grangier and A. Vinciarelli, “Effect of segmentation method on video retrieval performance”, *the Swiss National Science Foundation Res.Lab.*, Switzerland, CH-1920, 2005
- [9] G. Guo and S. Z. Li, “Content-Based Audio Classification and Retrieval by Support Vector Machines”, *IEEE Trans. on neural networks*, vol. 14, no. 1, January 2003
- [10] O. kalipzis, “Query Processing in multimedia database”, *Journal of applied science*, vol. 2, pp. 109-113, 2002
- [11] H. B. Kekre *et al.*, “Image Retrieval with Shape Features Extracted using Gradient Operators and Slope Magnitude Technique with BTC”, *International Journal of Computer Applications (0975 – 8887)*, vol. 6, no.8, September 2010

- [12] H. Kosch and M. Döller, “Multimedia Database Systems: Where are we now?”, Klagenfurt Univ., Dept. of Comp. Sci., 65/67, A -9020 Klagenfurt, Austria, 2006
- [13] H. Kosch *et al.*, “SMOOTH - A Distributed Multimedia Database System”, in *Proc. of the 27th VLDB Conf.*, Roma, Italy, 2001
- [14] G. Li and A. Khokhar, “Content-based Indexing and Retrieval of Audio Data using Wavelets”, Univ. of Delaware, Dept. of Elect. and Comp. Eng., Newark, DE 19716, 2000
- [15] W. Li *et al.*, “Facilitating Multimedia Database Exploration through Visual Interfaces and Perpetual Query Reformulations”, in *VLDB Conf.*, Athens, Greece, 1997
- [16] Z. Li, *Content-Based Audio Classification and Retrieval Using the Nearest Feature Line Method* (China, Microsoft Research, 2000)
- [17] L. Lord and C. Pratt, “Retrievals from DB2 BLOB (Binary Large Objects) Data Warehouse Using SAS”, Ph.D. dissertation, Vermont Univ., Microelectronics Division, Essex Junction, 2000
- [18] X. Ma *et al.*, “Content based Video Retrieval, Classification and Summarization: The State-of-the-Art and the Future”, 2010
- [19] C. Negoita, and M. Vladoiu, “Querying and Information Retrieval in Multimedia Databases”, *IEEE Trans. on neural networks*, vol. 8, no. 2, 2006, pp. 73-78, 2000.
- [20] B. V. Patel and B. B. Meshram, “Content based video retrieval systems”, *International Journal of UbiComp (IJU)*, vol.3, no.2, April 2012
- [21] M. Patella. “Similarity Search in Multimedia Databases”, Ph.D. dissertation, Univ. Degli Bologna, Dept. of Comput. Sci., 1999
- [22] T.C. Rakow *et al.*, “Multimedia Database Systems - The Notions and the Issues”, Technical Univ. of Darmstadt, Dept. of Comput. Sci., Technical Report D-64293 Darmstadt, Germany, 1999
- [23] C. Ribeiro and G. David, “A Metadata Model for Multimedia Databases”, Univ. of Porto, Dept. of Elect. Eng., Technical Report INESC Porto, 2004
- [24] V. Roth, “Content-Based Retrieval from Digital Video”, Inst. fur Graphische Datenverarbeitung, Technical Report D-64283 Darmstadt, Germany, 1999

- [25] H. Samet, "Techniques for Similarity Searching in Multimedia Databases", Center for Automation Research, Institute for Advanced Studies, Dept. of Comput. Sc., University of Maryland, College Park, MD 20742, 2008.
- [26] E. Saykoly *et al.*, "A Semi-Automatic Object Extraction Tool for Querying in Multimedia Databases", Bilkent Univ., Dept. of Comput. Eng., Sci. Report 85, Ankara, Turkey, 2003
- [27] D. Schonfeld and D. Lelescu, "VORTEX: Video Retrieval and Tracking from Compressed Multimedia Databases - Visual Search Engine", in *the International Conference on System Sciences*, Hawaii, 1999
- [28] D. Srivastava and Y. Velegrakis, "MMS: Using Queries As Data Values for Metadata Management", In *ICDE*, 2007.
- [29] A. de Vries, "Content and multimedia database management systems" Ph.D dissertation, Centre for Telematics and Information Technology, Univ. of Twente, Netherlands, 1999
- [30] J. Wu, "Content-Based Indexing of Multimedia Databases", *IEEE transaction on Knowledge And Data Engineering*, vol. 9, no. 6, November/December 1997
- [31] K. Wu *et al.*, "Notes on Design and Implementation of Compressed Bit Vectors", *Proc. of the 27th VLDB Conf.*, Roma, Italy, 2001
- [32] Y. Zhai *et al.*, "Video Understanding and Content-Based Retrieval", 2005