Issues in Querying Multi-media Databases

Chitta Baral, Graciela Gonzalez and Tran Son
Department of Computer Science
University of Texas at El Paso
El Paso, Texas 79968
U.S.A.
{*chitta,chelis,tson*}*@cs.utep.edu*

Introduction

The amount of information available to users, especially in the web, is growing exponentially month by month. To find useful information in the web, users normally have to rely on some sort of keyword matching searches, and/or navigate their way through the list of pointers to objects in different data repositories. This is not enough to satisfy the needs of the ever more sophisticated users. Keyword descriptions cannot effectively capture the contents of multi-media objects like text plus images, video or audio clips or a combination of them and does not allow sophisticated querying. To be able to pose and answer sophisticated queries we need multi-media databases. The special issue of IEEE Computer containing (GR95) and edited by its authors has several papers on multi-media databases involving images.

Our concern in this paper is querying multi-media databases. Even though, at the first glance multi-media databases seem to be very different from the traditional databases, in querying multi-media databases we would like to stay as close as possible to the approaches used in querying traditional databases. In the next section we discuss the similarity and differences between traditional and multi-media databases. We discuss the difference between databases and data repositories and show that the approach that has been normally followed to create traditional databases (from their repositories) can be used with some modifications for creating multi-media databases from multi-media repositories.

But we do have to pay special attention to certain complicated queries that are usually asked to multi-media databases. Consider for example a query to find, given a particular location on Earth, locations with a similar environment and climate; or a policeman who wants to find pictures of convicted felons that are similar to the picture (or frame image from a video) of the person who just robbed a convenience store; or an ornithologist who wants to find similar bird mating calls to establish a link between two species. These queries involve *similarity*, and thus surpass the capability of the traditional 'equality/inequality' and 'like' queries. Even the definition of similarity is not well established. In a later section we discuss the issue of 'similarity' between multi-media objects[1] and show how it can be used in queries.

Repositories vs Databases

A multi-media database is different from a multi-media repository, which is basically a collection of multi-media objects. Some examples of such repositories are:

- Collections of digitized pictures (perhaps with some key words and a caption); For instance the web site *http://www.cybercomm.net/ paltiel/CDRR/gadget _archive.html#DIGITIZED_IMAGES* is a repository of digitized pictures.
- Collection of movies; For instance the web site *http://us.imdb.com* contains a repository of movies.
- Collection of audio clips; For instance the web site *http://www.satchmo.com/nolavl/nomusic2.html* is a repository of audio clips.
- Collection of multi-media documents.

It is important for such collections not to be considered as multi-media databases, but only as repositories. The main difference is that our ability to query these 'collections' is very limited. *The first issue we would like to discuss in this paper is how to convert a multi-media repository to a multi-media database.*

To answer this question let us look back at conventional databases. We are all familiar with the student record database and the database maintained by a supermarket (Ull88). Actually, we are too familiar with them! So much that we do not often think about the 'repository' behind these databases. The repository behind a supermarket database includes sales receipts, purchase orders, invoices, employment applications, and many other pieces of information. *The main reason for having a database instead of simply storing the repository in electronic form, is to be able to answer more and more varied queries efficiently.* Actually, one of

[1] For lack of a better term, we use 'objects'. Our usage is not meant to advocate 'object-oriented databases'.

the basis of the design of any database is the types of queries that will be submitted.

On the other hand, we can almost always find information in the repository which cannot be obtained by querying the database, as that query was not considered during the design of the database. For example, when the IRS investigates a corporation, it looks beyond the database of the company, like at other computer files, invoices, account statements, reports, and other documents, to find information not in the database.

What we have tried to establish so far is that the fact that multi-media repositories are different from multi-media database is not novel and conventional database also differs from the repository behind it. This realization provides us with directions on how to proceed in creating a multi-media database out of a multi-media repository. The main lesson is that, as is normally done during the design of conventional databases, we should also consider the queries that our database is supposed to answer, when designing a a multi-media database that will contain information about a multi-media repository. Based on that, and after an appropriate data model[2] is chosen, we can further proceed in designing our database.

Suppose we decide to use relational data model. Then our approach would be to come up with relations, and their attributes and keys that we will use in our database. We would have done the same if we were designing a conventional database instead. The difference comes in when we start worrying about questions such as: What values an attribute of a relation may have? How are these values entered?

In case of a conventional database, values of attributes are normally character strings, or numbers and are usually entered by a data entry operator. Sometimes a document in the repository is computer readable (example: answer sheets of GRE exams) or in a format that makes it easy for a computer program to process and put in the database.

To elaborate on the additional concerns that arise when designing a multi-media database, let us consider the design of a database for the family pictures of the Kennedy family. Let us assume we would like our database to be capable of answering questions such as:

- List all pictures that contain Jack Kennedy.
- List all pictures where Jack Kennedy is in shorts.
- List all pictures where Jack Kennedy is to the right of Jackie Kennedy.
- List all pictures with Kennedy family members in a beach.
- List all pictures where someone from the Kennedy family is playing with a dog.
- List all Kennedy family members who are always with their pets in their pictures.

Our multi-media repository only consists of pictures with short captions, which may not always contain adequate information. Based on the above questions, we can come up with a relational data model. The next step would be to enter the data in the database. One way would be do have trained data entry operators who can quickly recognize the family members and the pets, and have them go through the pictures one by one and enter the data.

Now let us list the additional concerns that arises in creating a multi-media database from a multi-media repository.

- Data entry persons need to have added experience and qualification of quickly comprehending multi-media objects and extracting the necessary info from them.
- Programs for extracting necessary and useful info from a multi-media objects need to be more sophisticated than when dealing with conventional databases. For example, to automatically extract which pictures contain Jack Kennedy a program has to have sophisticated image processing and pattern recognition abilities.
- The values of some attributes could be multi-media objects (or fragments of multi-media objects). One way to do this would be to allow attributes to be pointers to files containing the multi-media object.

In the next section we give examples of creation of multi-media databases for two multi-media repositories.

Creating multi-media databases from repositories

Relational databases for different multi-media repository were created by groups of UTEP students in a database class in Spring 96. We will show two examples taken from these projects: a database created for the CNN repository in the web and another one for an image repository of Rotifers[3] in the Biology Department of UTEP.

The CNN Repository

The CNN repository can be thought of as a collection of documents identified by their URL address. Each document can then be divided into different sections. Each section will have its own title, abstract, keywords.

[2] At this point we are not concerned about whether relational data model or object-oriented data model or some other data model is more appropriate.

[3] Rotifers is a class of minute, many-celled aquatic animals having the anterior end modified into a retractile disc bearing circles of strong cilia, which, when in motion, look like rapidly revolving wheels.

Also, in each section, we can find text, images, sounds, and other multi-media data, or links to other sections or documents. Based on these observation, the following relational model was created for the CNN repository:

1. Document(URL)
2. Section (URL, LinkID, Title, Abstract, Keywords, Description)
3. Links(LinkID, Title, Object_Type, Object_URL)

By using an SQL–like query language in this model, we can answer more interesting queries than most (perhaps, all) search engines currently available on the internet. We would be able to search not only by keywords but also using more complicated expressions. For example,

1. The query "List all pictures of NBA games." can be expressed in SQL as:

   SELECT *Object_URL*
   FROM *Links, Section*
   WHERE *Links.LinkID = Section.LinksID*
   AND *Object_Type='Picture'*
   AND *'Basketball, NBA' IN Keywords*

2. Similarly, the query 'List all article about the Israel and Arab Peace accord." can be expressed in SQL as:

   SELECT *Object_URL*
   FROM *Links, Section*
   WHERE *Links.LinkID = Section.LinksID*
   AND *Object_Type='Text'*
   AND *'Israel,Arab,Peace Accord'* **IN** *Content(Object_URL)*[4]

Note that, using relational model and SQL we can ask even more complicated queries such as: "List articles that refer to articles about the the Israel and Arab Peace accord.", which can not be easily asked using current key word based search engines. In addition, the relational schema used in the above example can be easily modified to create a relational model for other WWW repositories.

'Rotifers' Image Repository

The next example shows how an SQL–like language that can return answers to queries posed to a multi-media database in a more interesting way.

The repository in this example contains information about *Rotifers* such as name, picture file, photographer' name, etc., available from the home page *http://www4.mmtlc.utep.edu/database/*. The repository was converted into a relational database using the following relational schema:

[4] This function loads the full content of the text file at the address pointed to by Object_URL

1. Rotifers(name, type, sizefile, filename, thumbname, height, width, art, author)
2. People(author, address)

Queries can be entered through a form available in the same home page. Again, an SQL–like query language was implemented for use in the search engine. A default method of returning the answer to a query is applied by displaying the picture associated with the record in the result table.

For example, the search engine returns the answer to the query

SELECT *name, filename, thumbname*
FROM Rotifers, People
WHERE *New York* **IN** *People.address*
in the following format[5].

Name	File name	Picture
Philodina roseola	ROSE.JPG	Pic.
Philodina roseola	ROSE2.JPG	Pic.
...	...	...
Adineta vaga	VEGA.JPG	Pic.

Similarity between objects

Queries that in a traditional database where handled routinely, like a pattern-matching query using the "LIKE" operator of SQL, turn extremely complicated when posed to a multi-media database. No longer are we trying to find similar pattern values in a character attribute, but similar pictures, sounds, or videos. How do we find convicted felons similar to a picture of a man that robbed a convenience store? How do we find similar sound patterns in a database of bird sounds? This section focuses on similarity queries: finding a set of (multi-media) objects "similar" to some other (multi-media) object.

But what exactly is similarity? In (JMM95) the authors point out that the meaning of similarity may vary depending on the application domain or the purpose of a query. And this is exactly what is observed by surveying the works of many authors that use the notion of similarity queries (Please see the references in (JMM95).). Each author works on a particular domain and uses one specific notion of similarity, usually tightly tied to the way data is represented.

As argued in (JMM95), an abstract and more general notion of similarity is necessary for uniform treatment in databases. The definition of similarity in (JMM95) states that "An object A is considered to approximate

[5] See more about this at the URL: http://www4.mmtlc.utep.edu/database/

an object B, if B can be reduced to A by a sequence of transformations".

However, we believe this notion is not general enough. Consider the example presented in (JMM95), where similarity queries are posed to a database containing a set of sequences (stock price data), each describing a real-valued function $f(x)$ on some real interval encoded as a sequence of real number pairs. The first pair specifies the x and y coordinates where the graph of the function starts, and each of the following s, l pairs represent a linear segment, where s is the slope and l is the x extent of the segment. The purpose of the query is to find functions that look like the series **A** whose function is depicted in Figure 1.

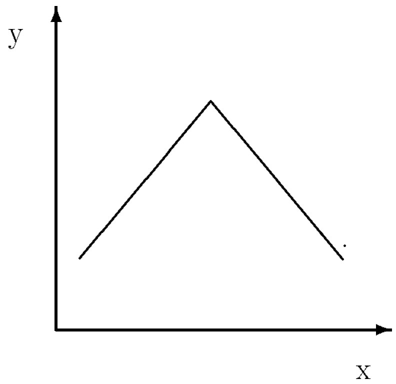

FIGURE 1.

After applying a refinement transformation rule that collapses adjacent segments $s_1 l_1$ and $s_2 l_2$ to a single segment whose slope is the weighted average of the two slopes and whose length is the sum of the two lengths, and according to their definition of similarity, the series **B** represented by the graph in Figure 2 is found to be similar to **A**.

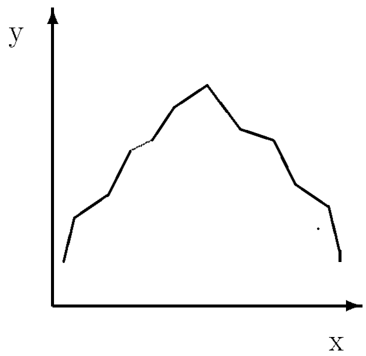

FIGURE 2.

Consider now the series **C** represented in Figure 3. Intuitively, one should expect B and C to be considered similar. However, if there is a general transformation that can be applied to **C** to reduce it to **B** or vice-versa, it is extremely hard to define, and it won't be a general transformation that could be applied to other cases. You could transform both to **A**, and thus find them similar to **A**, but no "similarity" would be established between **B** and **C** or between **C** and **B**.

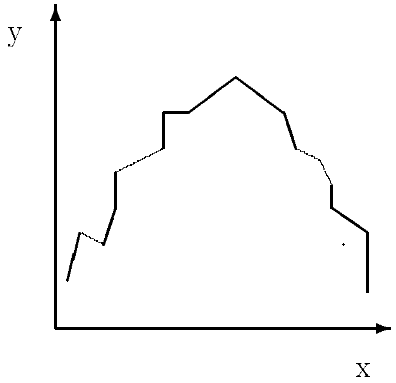

FIGURE 3.

What we need is a notion of similarity general enough to capture the similarity between B and C. We start by representing (JMM95)'s notion of similarity using a tree, as follows:

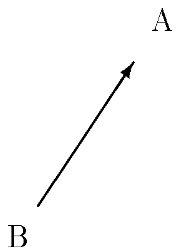

FIGURE 4.

The idea is that by applying transformation t to **B** we reduce it to **A**. Now we want to extend the graph as follows, to include **C**:

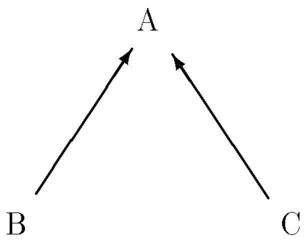

FIGURE 5.

A more general notion of similarity would allow us to express the relationship between **A**, **B** and **C** as follows:

- **A** generalizes **B** in 1 t-step.
- **B** is more specific than **A** in 1 t-step.
- **C** is more specific than **A** in 1 t-step.
- **B** and **C** are indirectly similar (or, are separated) by 1 t-step.

The above definition can be easily extended to more than 1 t-steps.

Our notion of similarity extends the definition of similarity in (JMM95). We now show how it can be incorporated into query languages, like SQL, to allow the expression of similarity queries. We use the flight examples from (GGM92) where the notion of similarity is implicitly used in giving co-operative answers. In our example, we use the notion of similarity explicitly.

For example, consider a database that keeps information about point-to-point (non-stop) flights between cities, using the following model:

FLIGHT(Airline, FlightNo, From, To)

Suppose, we have a database of airports and groups of airports, with similarity information as depicted in Figure 6. That is, using our definition given above, **Paris Airports** generalizes **Orly** and **De Gaulle**, **Orly** is more specific than **Paris Airports** by "location detail", and there is a first– level indirect similarity between **Orly** and **De Gaulle**.

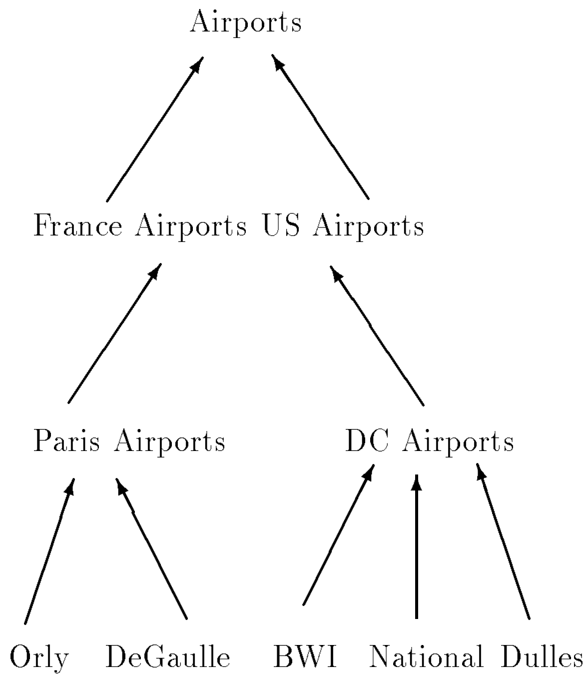

FIGURE 6.

If we allow the notion of similarity in and SQL–like query language, we could potentially make queries such as:

SELECT Airline, FlightNo
FROM FLIGHT
WHERE (PARIS_AIRPORTS GENERALIZES FLIGHT.FROM) **AND**
(DC_AIRPORTS GENERALIZES FLIGHT.TO)

to find out flights from Paris to Washington DC Airports, in general, and

SELECT Airline, FlightNo
FROM FLIGHT
WHERE (FLIGHT.FROM INDIRECTLY_SIMILAR "Orly" STEPS = 1) **AND**
(FLIGHT.TO INDIRECTLY_SIMILAR "National" STEPS = 1)

to find flights that go to an airport similar (1-step indirectly similar) to "National" from an airport similar to "Orly".

Acknowledgment

We acknowledge the help of students in the Spring 96 Database Class at UTEP (taught by Chitta) who created several multi-media databases using repositories from the web. We thank V. S. Subrahmanian for referring us to the article (JMM95). We also thank NASA for their grant NCC5-97, which enables Chitta to attend this workshop.

References

T. Gaasterland, P. Godfrey, and J. Minker. Relaxation as a platform for cooperative answering. *Journal of Intelligenet Information Systems*, 1(3,4):293–321, Dec 1992.

V. Gudivada and V Raghavan. Content-based image retrieval systems. *IEEE Computer*, Sept 1995.

H. Jagadish, A. Mendelzon, and T. Milo. Similarity-based queries. In *Proc. of PODS-95*, pages 36–45, 1995.

J. Ullman. *Principles of database and knowledge-base systems*, volume 1. Computer Science Press, 1988.