A Friendly and Intelligent Approach to Data Retrieval in a Multimedia DBMS

Daniel A. Keim[1], Kyung-Chang Kim, Vincent Lum

Department of Computer Science[2]
Naval Postgraduate School
Monterey, CA 93943 - 5100

Abstract

Manipulation of multimedia data is not straightforward as in conventional databases. One main problem is the retrieval of multimedia data from the database with the need to match the contents of multimedia data to a user query. In order to achieve a content based retrieval, in our approach, we use *natural language captions* which allow the user to describe the contents of multimedia data. In a similar manner, users will specify their queries on multimedia data contents in natural language form. A problem is that different or even the same user describe the same thing differently at different times which results in the descriptions of the contents of multimedia data to rarely exactly match the descriptions of the user queries. Hence, partial or approximate match between descriptions of multimedia data and user queries is generally required during multimedia data retrieval. We propose an intelligent approach to *approximate match* by integrating both object-oriented and natural language understanding techniques. In order to make the query specification process easier we also develop a graphical user interface supporting incremental query specification and a natural way of expressing joins. The Multimedia Database Management System (MDBMS) described in this paper incorporates the capabilities as mentioned above.

1. Introduction

A multimedia database management system supports the management of multimedia data, which includes image and sound among others, in addition to supporting conventional databases. Multimedia systems are currently gaining a lot of attention because technology today has made it possible to capture and store multimedia data in computers. Many applications like military, publishing or instructional routinely need multimedia data. Although the cost of the hardware required to handle multimedia data is decreasing rapidly, the software needed to manage such data is lacking or does not match the needs.

In this paper we present a Multimedia Database Management System (MDBMS). The system allows sophisticated handling of multimedia data featuring an intelligent data retrieval as well as a graphical interface for user interaction. Besides describing the overall system architecture the important parts of the system such as Parser, Matcher and Graphical User Interface will be presented in more detail.

One important achievement of the MDBMS system is the efficient method for the retrieval of multimedia data by way of inexact matching. In conventional databases, retrieval of standard numerical and alphanumeric data is handled by utilizing the *content* of the data. The fundamental problem that one must face in the context of a multimedia database is the question of how to provide content search. There is no easy solution. It is difficult to find the appropriate data conveniently and efficiently based on the contents of the multimedia data because they are intrinsically rich in semantics. In developing an efficient retrieval method for multimedia data, we concluded that it is not possible to utilize the content directly with today's technology since it is mostly unstructured complex data like an image or a sound.

In our MDBMS system we use the approach of content based search by means of verbal descriptions on the contents of multimedia data. We argue that the well known keyword approach to content description is not suitable because it has been known to be imprecise and the users often have difficulty in focusing the search to data of interest. Hence, we adopt the natural language approach to content description as a more viable option. The methodology we adopt consists of associating natural language captions to each multimedia data and using the description to retrieve the relevant data. More precisely, the description of a multimedia data is matched against the description of a user query which is also expressed using natural language captions. The major problem with this approach is that it is generally the case that the description of a multimedia data does not exactly match the description of a user query. The reason is that it is difficult for different users or even the same user at different times to describe the same thing identically because they can use synonyms, generalize/specialize categories and so on. Hence, the key to efficient retrieval is to automatically perform partial or approximate match of the description of multimedia data to the description of a user query whenever exact match is not possible. In this paper, we propose an intelligent approach to approximate matching by integrating object-oriented and natural language understanding techniques.

1. The author's permanent address is Institut für Informatik, Universität München, Theresienstr. 39, 8000 München 2.

2. This research was supported in part by NOSC, Direct Funding and the German Scholarship Foundation.

The second issue addressed in this paper regards new ways of interaction with the user. The user interface is an important part which strongly determines the effectiveness in using a system. In order to achieve a natural way of interacting with the MDBMS system we are developing a graphical user interface incorporating advanced features to make the query specification easier compared to query languages like SQL. We found that in order to formulate complex queries a user partitions it into smaller pieces and put them together at a later stage. This behavior is reflected in the principle of *incremental query specification* which is supported by our graphical user interface. In addition, we observed that, for a given database, the joins necessary to specify most of the queries directly correspond to natural language expressions. This leads to the principle of *natural expression of joins* also supported by our graphical user interface. Both principles are of general use not only for multimedia systems or graphical user interfaces but for any database query interface.

This paper makes three contributions. The first contribution is that context description of multimedia data is possible using natural language captions which can be interpreted automatically using domain dependent knowledge. The second contribution is the formulation of a general scheme to retrieve multimedia data with special emphasis on approximate match. As far as we know, very little research on partial or approximate matching has been conducted in the area of natural language processing. The third contribution is the identification and application of two principles in the construction of a graphical user interface that help to make the query specification process easier.

The paper is organized as follows: Section 2 discusses related work. Section 3 addresses fundamental problems and outlines the architecture of the Multimedia Database Management System (MDBMS). Section 4 describes the natural language interpretation capabilities of the parser, Section 5 the approximate match algorithm used for retrieval of multimedia data and Section 6 gives a short overview of the user interface. Finally, Section 7 gives the summary.

2. Related Work

Several multimedia projects have been undertaken by various researchers in both academia and industry over the past several years. The MINOS system [5] developed by a team at the University of Toronto manages highly structured multimedia objects. Sophisticated browsing and user interface features allow browsing of the schema as well as synchronized updates. The MCC database program [15] also undertook several multimedia projects and identified requirements for a data model suitable for multimedia applications. [10] has developed a framework to classify and compare the different multimedia projects.

The user interface is an important part of a database system especially when dealing with multimedia data because of their non-textual nature. Most of the research in the area of user interfaces focus on the entity-relationship [12] or the more complex sematic and object-oriented data model [1,3] allowing queries to be directly specified within the schema. In contrast, we use an extension of the relational model to handle and manipulate the media data. In order to allow an easy query specification we provide a graphical user interface which incorporates incremental query specification and a natural way of expressing joins, differing in many ways from the well know OBE interface [16].

Another important aspect of a multimedia database system is the content retrieval of media data. In [9], we introduced the approach of contents based search by means of natural language descriptions. This approach is related to research on artificial intelligence (AI) and information retrieval (IR). In the area of AI a variety of methods have been developed for the processing of natural language. [6] exemplifies the current state of the art. Most of the work done focus on complete understanding of natural language requiring extensive knowledge bases with general world knowledge. Our approach is somewhat simpler. We are only dealing with a subset of natural language being broad enough to allow a natural description of the media data but easier to understand than general language. Furthermore, we found that for most applications the knowledge base is domain specific allowing us to deal with a much smaller one for each domain. Both aspects contribute to an acceptable performance which is critical for a database system.

In the domain of IR there had been early interest in using AI techniques [14]. More representative of modern attempts are the IRUS system [2] which is designed for processing heterogeneous data bases through natural language queries and the IOTA system [4] which tries to improve the qualitative performance of IR systems in replacing keywords by noun groups. The approach we propose is somewhat different from the intelligent IR systems proposed so far. It is clear that most of the work in these systems is mainly concerned with natural language processing, particularly query processing, and deductive capabilities based on an extended semantic model of document content. Our approach also shares these characteristics. However, in most systems the concept of matching between system and user concepts is based on exact matching while our approach is based on approximate matching. Even in systems with approximate matching capabilities, the matching function used is primitive or superficial at best compared to our approach which integrates object-oriented technology and natural language understanding to improve the quality of the matching.

3. Architecture of Multimedia Database Management System (MDBMS)

In this section, we outline the architecture of the MDBMS. The architecture consists of the various components of the MDBMS system. Before we continue, definitions and various issues associated with the data model used in the MDBMS system are addressed.

3.1 Definitions and Background

As mentioned before, multimedia data consists of unformatted data such as text, image, voice, signals, etc. in addition to alphanumeric data. We define a multimedia database management system (MDBMS) as a system that manages all multimedia data and provides a mechanisms to handle concurrency, consistency, and recovery in addition to providing a query language and query processing.

Despite differences in data model and implementation aspects, all research projects on MDBMS have decided to organize multimedia data using the abstract data type (ADT) concept. This is generally accepted as the adequate approach. However, none of the projects have addressed the problem of content retrieval of multimedia data.

The fundamental difficulty in handling multimedia data is intrinsically tied to its very rich semantics. To illustrate the difficulty, let us look at an image of ships. How are we to know what type of ships are in the picture? As another example, let us suppose that there is a picture of a dog and a cat. How do we know if they are chasing each other or playing? To answer queries posed on images, for example, a person must draw from a very rich experience encountered in life to derive at a good answer. One must have a sophisticated technique to analyze the contents of the images to get the semantics of the different things in the images. Technology today is not advanced enough to expect systems to have this kind of capabilities to answer multimedia queries. However, we can use both AI and IR technology to do the next best thing. We can abstract the contents of multimedia data into words or text and use the text description equivalent of the original multimedia data to match a user query. This principle we will use in designing a MDBMS to handle multimedia data for different applications.

Figure 1 shows the format of a multimedia data which consists of the registration, raw and description data. Raw data is the bit string representation of the image, sound, signal, etc. obtained from scanning or digitizing the original multimedia data. Registration data generally enhances the information about raw data and is not redundant. The contents of a multimedia data is described by the description data. Description data cannot be automatically derived by the computer given the technology today. We assume that users will supply the description data for multimedia data in a natural language form.

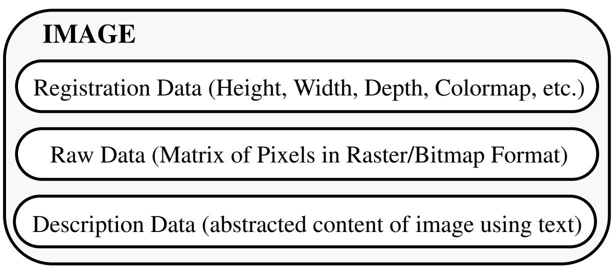

Figure 1: Example for the Multimedia Data Format

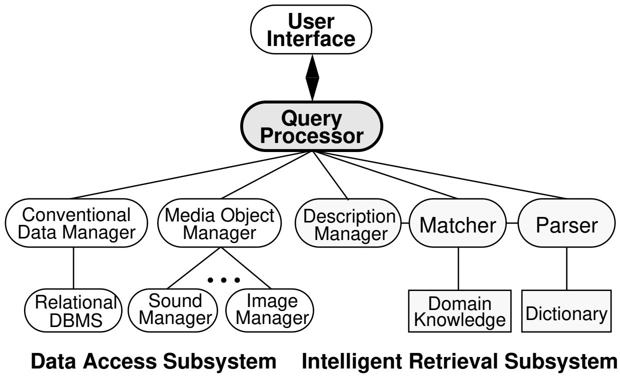

Figure 2: Architecture of MDBMS System

3.2 Architecture

In this section, we present the various components of our MDBMS. This is a modified version of the architecture of the MDBMS discussed in [9]. Our proposed architecture enhances the performance of the matcher component and adds the capabilities of the user interface which are lacking in the architecture proposed in [9].

As shown in Figure 2, the components break down into user interface, query processor, data access and intelligent retrieval subsystem. The data access subsystem consists of conventional and media manager and controls the access to the actual data stored in relational and media DBMS. The intelligent retrieval subsystem is composed of parser, generator, matcher and description manager. The query processor accepts queries from users and executes them by calling the other components. When a new description for a multimedia data is entered, for example, the query processor calls the parser. The parser uses the dictionary to produce first-order predicates and returns them to the query processor. The query processor then hands the predicates over to the description manager which links the description to its multimedia data.

When the query processor receives a query the first task is to decompose the query into subqueries affecting only conventional or media part. The conventional subquery is passed to the conventional data manager without modifications. For the text description, the query processor calls the natural language parser to obtain the equivalent query predicates. The predicates are then handed to the matcher. The matcher tries to match the query with multimedia data by comparing the predicates of the query with that of the stored multimedia data. The matcher does this by calling the description manager and using domain knowledge. In addition, if an exact match is not possible, the matcher automatically switches to approximate match. As the solution to the natural language part of a query, the query processor receives links to the qualified multimedia data. After combing them with the results of the conventional subquery the final results are retrieved by the Data Access Subsystem.

The query processor, conventional and media object manager, description manager, parser and matcher have already been implemented as part of the MDBMS prototype system developed at the Naval Postgraduate School [7,9,11].

4. Natural Language Understanding in the Parser

In this chapter we describe the natural language understanding capabilities of the parser. We outline that in order to accomplish the goal of content retrieval of multimedia data, full understanding of natural language is not necessary. However, a restricted interpretation is necessary which is done by the parser component using the application dependent dictionary as a semantic basis.

4.1 Natural Language Descriptions

As mentioned, we propose to perform retrieval of multimedia data by matching the natural language descriptions with query specifications. We believe that unrestricted natural language processing is very difficult to achieve given the AI technology today. We found that the language needed to describe multimedia data is much more formal than everyday English. Hence, instead of natural language description, we use captions to describe multimedia data. Captions are a natural but special, stylized way of writing descriptions with a subset of natural language and not as difficult to parse and interpret as general natural language.

Additionally, for a particular multimedia application the universe of discourse is usually quite constraint. Nouns tend to be concrete and most multimedia databases emphasize still photographs and other fixed time graphics to which few verbs can be applied thereby easing a difficult aspect of natural language processing. Important is that we use natural language only to access entities in a database making complete understanding unnecessary. The details of captions and their restrictions for our objectives are beyond the scope of this paper and are given in [7,13].

4.2 Dictionary

Besides the captions themselves, our system requires auxiliary information from a dictionary. The dictionary or lexicon is necessary for parsing and gives each possible natural language word its semantic: its part of speech, its grammatical form and the form of literals needed to represent it. Many of the words - for example, conjunctions and qualifying adjectives - are consistent in meaning across a wide range of domains; thus we can borrow their interpretation from existing natural language systems and include them in our dictionary. The words that significantly change between applications are nouns and few verbs. They need to be defined for every application domain separately, but mostly their meaning is straightforward. To simplify matching, we are trying to limit the properties and relationships to a small set of primitives, for example we will not distinguish between the relationship asserted by the terms 'within', 'inside', 'part of', 'containing' and 'comprising'. This can be done without loss because in order to achieve efficient retrieval it is not necessary to capture the full meaning of an English expression, but just the main intent.

4.3 Natural Language Interpretation

The parser translates the text description into a set of predicates called *meaning list*, thereby reducing imprecision and ambiguity of the natural language descriptions considerably. These predicates state facts about the real world entities involved with multimedia data like their properties and relationships. As in most parsing methods, we chose the use of first-order predicate calculus as a formal representation of the description data. The parser depends on the dictionary to turn the descriptions into predicates. It is the parser's task to use the dictionary to resolve synonyms and to check the syntactic context to resolve lexical ambiguities.

An example of a natural language description and its translation into an equivalent set of predicates is shown below.

Description: "*A car with red body*"

Predicates: *car(x), component(x,y), body(y), color(y,red)*

Our parser also provides a mechanisms to automatically partition a query into subject, verb and object components. This is essential in that, during data retrieval we use the partitioned components to match against domain-dependent knowledge which also break down into subject, verb and object categories. Other important features of the parser are the use of *supercaptions*, a generalization of captions, and *frames for stereotypical actions*, allowing a set of predicates to be derived from terms in the description.

Our current implementation of the parser uses augmented-transition network parsing and interpretation routines. It is implemented in Quintus Prolog and running on a SUN SPARC workstation. The details of parser and predicates are beyond the scope of this paper and are given in [9,13].

5. Matching

In this chapter, we propose new ways of matching natural language descriptions of multimedia data with query specifications. The key to our matching process is the use of domain knowledge represented using the notion of class hierarchy borrowed from the object-oriented field.

5.1 Domain-Dependent Knowledge

To represent domain-dependent knowledge, we chose the object-oriented data model. The object-oriented model supports highly structured, complex objects and can capture

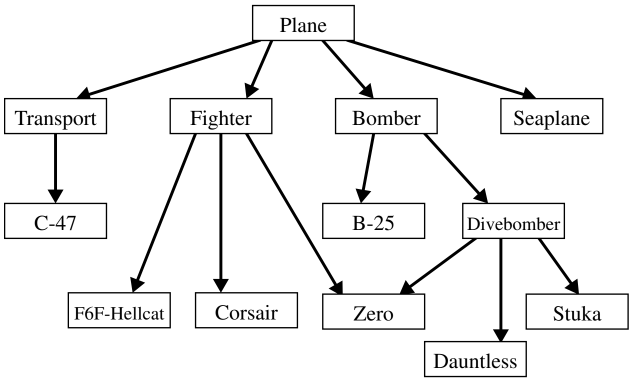

Figure 3: Generalization Hierarchy of a Plane

naturally any mini-world entity. It supports generalization and specialization abstraction which permits conceptual generalization on the contents of the captions. Researchers [7,15] have identified the use of the object-oriented model in multimedia database applications as an appropriate and viable option. Without loss of generality, in this paper we will restrict our domain of discourse to the domain of the military history of US forces in the Pacific during World War 2. The main reason is that we tested our current prototype MDBMS in a military application based on the domain of the US military history. For our purposes, we will apply the approximate matching technique to this domain; however, we claim that it can be applied to other multimedia applications as well.

Figure 3 shows an example of the generalization hierarchy of plane, a noun concept in our domain of discourse. It is the domain-dependent knowledge on planes that participated in the Pacific during World War 2. We assume that the reader is familiar with object-oriented concepts such as object, class, inheritance along class hierarchy or lattice and methods. We also assume that the direction of the arrow in Figure 3 is from a class to its subclass.

5.2 Partial Matching Algorithm

In this section, we will discuss our partial matching algorithm by following through an example. Suppose that we have images of planes stored in the multimedia database and the images are described as transport planes. Let us now assume that a user gives a query asking for all planes which are C-47s. Even though there is no exact match, we should retrieve all transport planes stored because any C-47 is a transport plane according to the domain-dependent knowledge. Now, if the user asks for all fighter planes, we cannot simply retrieve all transport planes because they may not be what the user wants. However, a user asking for planes would more likely retrieve the stored transport planes than if he was to ask for fighter planes because a transport plane is still a plane but is not a fighter plane.

The goal of our algorithm is also to minimize the influence of the definition of the hierarchy which depends on the designer. The generalization hierarchy designer might have a view of the domain dependent knowledge which may not be consistent with the view of other people. This phenomenon might bias some specific branch of the generalization hierarchy over other branches during partial matching. Hence, our major objective is to come up with a weight ranking scheme that is both fair and accurate which can be used to determine whether stored multimedia data should be retrieved given a user description.

5.2.1 Weight Ranking within a Class Hierarchy

The weight ranking strategy used is a consequence of the semantics of the class hierarchy (lattice) or the IS-A hierarchy concept supported by the object-oriented data model. Given a class C in a generalization (class) hierarchy for a noun or a verb concept, we can introduce the following two general heuristics:

Heuristic 1: *All subclasses of C (specializations) have positive weights.*

Heuristic 2: *All superclasses of C (generalizations) have negative weights.*

Heuristic 1 says that given a class C specified in a user query, all subclasses of C in the class hierarchy to which C belongs are specializations of the class and more weights (positive) are given. Heuristics 2 says that given a class C specified in a user query, all superclasses of C in the class hierarchy to which C belongs are generalization of the class and less weights (negative) are given. This reasoning follows directly from the definition of a class (IS-A) hierarchy and relationships among classes along the class hierarchy in the context of an object-oriented data model.

The assignment of negative weights to generalization is intuitively clear. The assignment of positive weights to specialization is based on the fact that specialization inherits all properties of the parent nodes in addition to having its own additional information. Hence, we feel that positive or more weights should be assigned to the nodes in the paths towards specialization hierarchy.

In the following, we discuss the weight ranking algorithm resulting from the heuristics. There are three different situations in which weights can be assigned to classes in a class hierarchy (see Figure 4). Suppose that the class specified in a user query is class C. As before, we assume that the direction of the arrow is from a class to its subclass. The first situation, shown in Figure 4(a), is to assign weight to a class

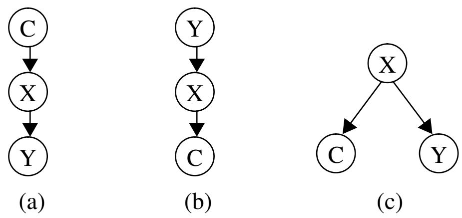

Figure 4: Three Situations in a Class Hierarchy

(X or Y) which is a subclass of class C. The second situation, shown in Figure 4(b), is to assign weight to a class (X or Y) which is a superclass of class C. The third situation, shown in Figure 4(c), is to assign weight to a class (Y) which is a subclass of a superclass (i.e. X) of class C.

The assignment of weights for the three situations is done as follows: First, we assign a weight of *0* to the class C specified in the user query. Class C is the reference point to all other classes in the class hierarchy during approximate matching. For classes which are subclasses of class C, we assign *positive weights* because they are specialized version of class C. Specialized versions of class C have more specific and definite information than C itself and hence are assigned positive weights instead of 0. For our purposes, all subclasses of C are assigned the same positive weight.

For classes which are superclasses or subclasses of superclasses of class C, we assign *negative weights* because they are generalized version of class C. Generalized versions of class C have less and more general information than C itself and hence are assigned negative weights. Different generalization versions have different negative weights. However, in assigning negative weights, we have to minimize the influence of the definition of the model. It is true that the further away a class is from class C in the class hierarchy, the more negative weight is assigned to the class.

In most systems, the assignment of weight of a class is linearly inverse proportional to the depth level of the class relative to the level of the class C specified in the user query. We believe that this is not the correct approach because the relative distance of a particular class to the class of interest, in this case class C, with respect to other classes is not the absolute but some artificial distance caused by a particular designer's view of the domain knowledge. The main problem with this approach is that some classes belonging to some lengthy branch could be unfairly disqualified because of higher negative weights. Our weight ranking system tries to minimize the bias against some lengthy branch of a class hierarchy over other shorter branches.

Given that class C is the class specified by user query as shown in Figure 4, the assignment formulas of weights for classes in a class hierarchy according to the three different situations mentioned are as follows.

(1) Class specified by user query (i.e. class C in figure 4)

$$\mathbf{weight} = 0$$

(2) Subclass of class C (i.e. class X or Y in 4(a))

$\mathbf{weight} = \alpha$, where α is a integer constant

(3) Superclass of class C (i.e. class X or Y in 4(b))

$$\mathbf{weight} = -\left(\alpha \times \sum_{i=1}^{n} \left(\frac{1}{\beta}\right)^{i}\right),$$

where α, β are integer constants and n is level # of superclass relative to class C

(4) Subclass of a superclass of class C (i.e.class Y in 4(c))

$$\mathbf{weight} = -\left(\alpha \times \sum_{i=1}^{h} \left(\frac{1}{\beta}\right)^{i}\right) - \left(\gamma \times \sum_{j=1}^{l} \left(\frac{1}{\upsilon}\right)^{(l+1-j)}\right),$$

where $\alpha,\beta,\upsilon,\gamma$ are integer constants; h is level # of superclass relative to class C and l is level # of subclass relative to superclass

In our scheme, a class which is assigned a positive weight is always selected during partial matching. A class with a negative weight can be selected provided that it does not exceed a threshold value set by the user.

To understand the weight assignments for different classes, we next give some examples using the class hierarchy of Figure 3. Given a user query, if the image corresponding to the user description is not found in the database, the system automatically proceeds with approximate matching. For a given query, the weight assignment formulas are used to calculate the weights for some of the classes in the class hierarchy. For the sake of argument, we assume that the values of α, β, γ and υ are 40, 2, 48 and 2 respectively.

(1) "**A transport plane sank in the Pacific**"

Transport = 0, C-47 = 40, Plane = -20,
Fighter = - 44, Corsair = -56

(2) "**A F6F-Hellcat sank in the Pacific**"

F6F-Hellcat = 0, Plane = -30, Seaplane = -54,
Stuka = -72, C-47 = -66

(3) "**A bomber sank in the Pacific**"

Bomber = 0, Stuka = 40, B-52 = 40, Plane = -20,
Seaplane = - 44, C-47 = -56

In the examples shown, all classes which are assigned positive weights are selected during partial matching. In example (1), the positive weight of 40. for the class C-47 means that the image whose description is *"A C-47 sank in the Pacific"* is selected during partial matching. As shown in the examples, all classes which are subclasses of the class which is specified in a user query are assigned positive weights. All classes which are superclasses or subclasses of the superclasses of the class which is specified in a user query are assigned negative weights. For these classes, the weight of a class is inversely proportional to the depth level of the class relative to the level of the class specified in the user query along the class hierarchy although they are not strictly linear. In example (2), the class Seaplane has a negative weight of -54. This means that the image whose description is *"A Seaplane sank in the Pacific"* has a weight of - 54. Class Stuka has a negative weight of -72 and class Stuka is further away from F6F-Hellcat than class Seaplane is from F6F-Hellcat.

Suppose the weight of a class is linearly but inversely proportional to the depth level of the class relative to the level of the class C specified in the user query. If we assign a negative constant weight, say -10, for each level away from class C, the class which is 5 levels away from class C will have a negative weight of -50 compared to a negative value of -20 for a class which is 2 levels away from class C. For example, if the user query is "**A Transport sank in the Pacific**", the weight of class Transport is 0, class Seaplane is -20 and class Stuka is -40. Using our formulas, the same user query will assign weights of classes Transport, Seaplane and Stuka to be 0, -44 and -62 respectively. The weight of class Stuka is more biased against relative to the weight of class Seaplane using the linear method over our method.

It is very difficult to quantify how much closer class Seaplane is to class Transport over class Stuka to class Transport as both Seaplane and Stuka are types of planes. Our formulas are designed to minimize bias as best as possible. A user is more likely to select a threshold value such that class Stuka is less likely selected during approximate matching over class Seaplane using the linear method compared to using our dynamic method. Another difficult task is to set the value of the constants to be applied in our assignment formulas as well as the threshold value. The user must choose the correct values for the constants and the threshold value depending on the number of objects that qualify during approximate matching. Hence, it is necessary for the system to interact with the user through the user interface throughout the matching process.

5.2.2 Weight Ranking for a Group of Class Hierarchies

In this section, we extend the ranking of weights for classes belonging to different generalization hierarchies. For each class hierarchy selected by the user query, the classes within the class hierarchy are assigned weights using the weight ranking system discussed in Section 5.3.1.

The global ranking of weights for different class hierarchies is a problem because the weights assigned within each class hierarchy now have to be considered with respect to weights assigned for other class hierarchies and they have to be meaningful globally. Only the user is able to determine the priority order of importance of the class hierarchies because it depends on his intent. To combine independently calculated weights of two class hierarchies (CH) the following weight formula can be used.

$$\textbf{Weight} = \alpha\ \textbf{(Weight (CH1))} + \beta\ \textbf{(Weight (CH2))}$$

The constant values α and β determine the priority order of CH1 and CH2. They are to be specified by the user. However, if the weight of CH1 or CH2 is zero, the constants α and β are set to one; hence, no information from the user is needed in this case.

For a practical system, the number of class hierarchies involved in the weight assignment is obviously large since many noun and verb concepts are involved. It is not difficult to see that our weight ranking scheme can be easily extended to assign weights for classes involving many class hierarchies.

5.2.3 Application of Weight Ranking Algorithm

The application of the weighting algorithm just presented requires a parser to understand the natural language specifications in the multimedia data descriptions and the user queries. As stated earlier, the descriptions are parsed and stored in the system as predicates. The queries are processed as follows.

When a query is received from the user, the parser separates the natural language specification into smaller component groups, namely subject noun, verb and object noun phrases. Each of these will actually become predicates. When these predicates match exactly with the predicates in the descriptions of certain multimedia data, those multimedia data will be retrieved. However, there may be other descriptions of multimedia data that are actually of interest to users but those descriptions are not stated as logically implied by the query. This latter category is expected to be the usual case rather than the former for reasons stated earlier.

To find the latter, we suggest that the system searches in the noun and verb generalization hierarchies of the object classes and assigns weights to the descriptions as given in the weight assignment algorithm, using the appropriate weighting factors (ω and δ in the previous section) as received from the user. Only the multimedia data with combined weight exceeding the threshold value set by the user will then be retrieved.

The separation of the natural language query can be in smaller components than the three groups just stated. For example, a complex noun phrase may be separated into a number of small noun groups and the weighting algorithm is applied to these groups to obtain a combined weight. Naturally, the finer the granularity of the separation, the larger and the more complex the processing is needed.

6. Graphical User Interface

The goal of a graphical user interface is to support the query specification process allowing the user to efficiently use the database system. It should allow inexperienced users to retrieve data from the database without having to know a specific query language. In today's database management systems the user is forced to think in terms of data model and query language, differing a lot from his way of thinking. Often a user can express a query easily in natural language, but has difficulties to express it in some given query language.

Most queries involve both media and formatted data. For the media part of the query we use our intelligent matching algorithm which is directly processing natural language captions. For conditions on formatted data, natural language expressions are mostly too imprecise to be directly processed. We try to overcome this problem by providing a graphical user interface supporting a natural query specification.

The data model adopted in our system is an extended relational data model. Despite some drawbacks the relational model has great advantages: It is well known, widely used and has a firm theoretical basis. For our purpose, we extend the relational model to capture media data types and, as shown below, we also extend the query language to allow the manipulation of media data and facilitate the query specification process.

Before describing the user interface of the MDBMS system, we first outline ways to achieve a natural query specification process.

6.1 Towards a Natural Query Specification

Usually, every user can describe a query (or at least the desired result) easily in natural language. Unfortunately, natural language expressions representing a query are imprecise and difficult to automatically translate into a formal query language to be understood by a database management system. We argue that the gap between the user's way of expressing a query in natural language and database manipulation languages like SQL can be improved considerably.

When comparing the user's natural language (NL) expression for a query with corresponding SQL statements the first difficulty is that the table and attribute names do not exactly match. In a graphical user interface this problem is easy to overcome. All table and attribute names can be presented to the user who simply selects the desired ones using a pointing device (e.g. mouse).

Another difficulty is related to joins between tables. Mostly the join condition is hidden in the user's NL expression. In examining a large number of queries expressed in natural language as well as SQL we found that, in most cases, the join condition directly corresponds to some specific NL expressions. Additionally, the number of joins used in most of the queries was small compared to the number of possible joins. This can be explained by two facts. First, the number of semantically meaningful joins is restricted and second, some of the most frequently used joins are already intended at the design time of the database. In order to provide a natural way of expressing joins, in our system we allow database designer and user to define and name joins prior to its actual use. A predefined join can involve more than two tables (e.g. two tables are joined by means of a third table) thereby providing a simple way of expressing m:n relationships. Once defined and named, all predefined joins can be used to specify a query. Predefined joins differ from views: First, the result of a predefined join is not a table as in the case of a view but a specific connection between tables. Second, predefined joins allow connections between different levels in nested queries and even recursive joins can be expressed. An example for predefined joins is given in the next section.

Another thing we learned in examining the process of query specification is the handling of complex queries. Given a complex data retrieval task the user partitions it into smaller subtasks which are easier to handle. Starting with the clear parts of the query the user deals with all parts and combines the results into the final solution. In our system we support this way of handling complex queries by an incremental query specification to be described in the next section.

Finally, we observed that a special category of queries is easy to express in NL but rather complicated in a formal query language. Additional operators, closely related to corresponding NL expressions, allow an easier and clearer query specification. Considering for example a query like *'Select the name of planes which can carry all weapons of the category air-to-air'* we found that a special *'all'* operator greatly enhances the readability and understandability of the SQL-like query making it similar to the user's NL expression. For the example, we presume to have the tables *plane*, *weapon*, *plane_weapon* and a predefined join named *carries* expressing the m:n relationship between planes and weapons.

```
select p_name from plane
where plane carries weapon
      and w_nr = all ( select w_nr from weapon
                       where category = 'air-to-air')
```

SQL statements expressing the same query without the all operator are rather complicated. Two possibilities are:

```
1. select p_name from plane
   where ((select w_nr from plane_weapon A
          where plane.w_nr = A.w_nr)
            contains
            (select w_nr from weapon
             where category = 'air-to-air') )

2. select p_name from plane
   where not exists
         (select * from plane_weapon B
          where B.w_nr in (select w_nr from weapon
                           where category = 'air-to-air'))
                and not exists
                    (select * from plane_weapon C
                     where C. p_nr = B.p_nr
                           and C.w_nr = B.W_nr) )
```

6.2 Description of the Graphical User Interface

In this paper, we will give a general idea of our graphical user interface by presenting a small example of the retrieval process. Due to space limitations we will only describe a small fraction of its capabilities.

After selecting the database the user gets the system menu providing the main database manipulation functions: insert, delete, update or retrieve (see figure 5). When selecting retrieval, the user gets the query specification window and his first step is to select the tables to be used in the query. For each selected table a list with all attributes will be displayed in a separate window and all predefined connections involving at least one of the selected tables will appear in the *Connections* window. To specify the result list (projection) the user has to move the desired attributes to the *Result List* window. Now only the conditions need to be specified. Using connections, attributes of the selected tables and operators provided by the *Tool Box* the query can easily be built using the mouse. In the *Query Representation* window the query is displayed graphically. Each part of the query is represented by a small box, simple conditions by a single, subqueries by a double box, and the connection lines are labeled with the kind of connection used. An advantage is that every part of the query can be addressed for edit or delete at any time during the query specification process. To enhance the clarity of display parts of the query can be grouped together and displayed as one box (zoom in). If the user wants to see the query in full detail at a later stage he can use the zoom out option.

To support incremental query specification we allow the user to start with any part of the query and combine the separate parts at a later stage. Additionally, we provide an option to save and reload any part of the query for later use.

Another important part is the way of specifying the natural language description part of a query necessary when media data are involved. If the user selects a media attribute in the specification of the condition, automatically a special description editor will be displayed in a separate window where the media description can be specified. The description editor has special features including buttons to check the description, present the hierarchy for a word and enter the weight of the different parts of the description needed for the approximate matching.

To get an impression of the user interface, the specification for the query *'Select the name, air base and image of planes which can carry all weapons of category air-to-air and where the image shows "plane attacking a hostile plane" '* is displayed in figure 5.

For the representation of results we choose a combined form and list oriented approach. Generally, the results are presented as a list. Media attributes are represented as buttons allowing to access the media data. By clicking to a row of the list a single tuple can be obtained in a form. Due to space limitations we can not elaborate on these features.

In this paper, we presented only a small part of our graphical user interface. The data definition, insert, update and deletion operation, query processing and optimization issues, representation of results, predefined joins, special operators and their semantics are far beyond the scope of this paper and will be presented in a later paper [8].

7. Summary

A major problem faced in a multimedia database system is the retrieval of multimedia data such as a sound or an image. Media data is intrinsically rich in semantics and conventional search methods used in databases and information retrieval systems may not work or are of little use. Most research on intelligent IR systems are concerned with natural language processing and deductive capabilities based on extended semantic models of document content. However, most of them deal with exact matching or primitive partial matching using simple linear methods.

Another problem faced in today's database systems is the lack of a natural way to specify complex queries. It is caused by the gap between the user's way of thinking and the query languages used in most systems. Although a lot of work has been done in the area of user interfaces for database systems no query language comes close to the natural query specification process used by humans.

In this paper, we discussed these fundamental problems and outlined the architecture of our MDBMS system. One contribution of our paper is the formulation of a partial matching algorithm that uses domain knowledge, represented using an object-oriented data model, and a weight ranking system to assign weights to different multimedia data stored in a database and selects those multimedia data that partially match a given user query description. Our parser, unlike others, provides an interpretation of natural language descriptions needed to achieve an intelligent retrieval of multimedia data. Further research is necessary to improve the parser to also automatically derive adjectives and other caption components for complete understanding and processing of captions in the context of partial matching.

A second contribution of this paper is our graphical user interface. It shortens the gap between the user's way of thinking and formal query languages by using graphical user interaction. In our system, we support incremental query specification, predefined joins and special operators to make the query specification process user friendly. The user is guided as much as possible allowing a quick and almost faultless query specification. Further research is necessary to come even closer to the user's way of query specification e.g. by allowing the user to directly communicate with the system in natural language.

We believe that our system provides a simple and elegant approach to both retrieval of multimedia data and query specification. The simplicity of our retrieval method lies in exploiting the semantics of generalization and specialization abstraction of the object-oriented model; the simplicity of the user interface lies in the natural way of query specification being directly obtained from queries expressed in natural language.

We also believe that our approaches are general ones that can be readily applied to other areas. Our retrieval method can be used for other applications in IR and AI and the ideas of our user interface can be easily applied to most database query interfaces.

References

[1] Agrawal, R. et al. "OdeView: The Graphical Interface to Ode", Proc. ACM-SIGMOD 1990 Int'l Conf. on Management of Data, Atlantic City, pp. 34-43, 1990.

[2] Bates, M. and B. Bobrow, "Information Retrieval using a Transportable Natural Language Interface," Proc. of the 6th. ACM SIGIR Conf. on R&D in Information Retrieval, Bethesda, MD, 1983.

[3] Bryce D. and Hull, R. "SNAP: A Graphics Based Schema Manager", Proc. IEEE Int'l Conf. on Data Engineering, Los Angeles, CA, pp. 151-164, 1986.

[4] Chiaramella, Y. and B. Defude, "A Prototype of an Intelligent System for Information Retrieval: IOTA," Information Processing and Management, vol. 23, no. 4, pp. 285-303, 1987.

[5] Christodoulakis, S. et. al., "Multimedia Document Presentation, Information Extraction, and Document Formation in MINOS: A Model and a System, " ACM TOOIS, vol. 4, no. 4, pp. 345-383, 1986.

[6] Grosz, B.J. et. al., "TEAM: An Experiment in the Design of Transportable Natural Language Interfaces," Artificial Intelligence, pp. 173-243, 1987.

[7] Holtkamp, B. et. al., "Demon- A media Object Model incorporating Natural Language Descriptions for Retrieval Support," Tech. Report NPS52-90-019, Naval Postgraduate School, Monterey, CA, 1990.

[8] Keim, D. A. and Lum, V. "A Graphical Database Interface for a Multimedia DBMS Supporting a Natural Query Specification Process", in preparation

[9] Lum, V. and K. Meyer-Wegener, "A Multimedia Database Management System Supporting Content Search in Media Data," Tech. Report NPS52-89-020, Naval Postgraduate School, Monterey, CA, 1989.

[10] Masunaga, Y. "Multimedia Databases: A Formal Framework," Proc. IEEE CS Office Automation Symp. IEEE CS Press, Washington, pp. 36-45, 1987.

[11] Meyer-Wegener, K. et. al., "Managing Multimedia Data," Tech. Report NPS52-88-010, Naval Post-graduate School, Monterey, CA, 1988.

[12] Rogers, T. R. and Cattell, R. G. G. " Entity-Rela-tionship Database User Interfaces" in Readings in Database Systems, edited by M. Stonebraker, 1988.

[13] Rowe, N. and E. Guglielmo, "Exploiting Captions for Access to Multimedia Databases," to appear in IEEE Computer, 1991.

[14] Spark, K. Jones, "Artificial Intelligence: What can it offer to Information Retrieval," Proc. of the Informatics 3, Aslib, ed., London, 1978.

[15] Woelk, D. and W. Kim, "Multimedia Information Management in an Object-Oriented Database System," Proc. of VLDB, Brighton, England, 1987.

[16] Zloof, M. M. "Query-By-Example: A Data Base Language," IBM Systems Journal, 4, pp. 324-343, 1977.

Figure 5: Screen after Specifying the Query