A Query Language and Interface for Integrated Media and Alphanumeric Database Systems*

Jia-Ling Koh*, Arbee L.P. Chen, Paul C.M. Chang, James C.C. Chen

Department of Computer Science
National Tsing Hua University
Hsinchu, Taiwan 300, R.O.C.
Email : alpchen@cs.nthu.edu.tw

Abstract. In this paper, we consider a multidatabase system which consists of media databases and traditional *alphanumeric databases*. The *semantic relationships* which may exist among the objects in the media databases and alphanumeric databases are introduced. By applying the semantic relationships, the related information in the system can be integrated. Therefore, more powerful querying capabilities and a higher degree of information sharing can be achieved. A query language IMAQL is designed for querying the integrated system. A graphic query interface is also presented to facilitate user-friendly querying in the system.

1 Introduction

Due to rapid development of electronic devices and computer technologies, more and more information is represented in various types of media. Databases for managing media data have been individually developed, including image databases, audio databases, and video databases [1] [4] [7]. Content-based retrieval of media data has become an important research issue. [17] provides strategies to detect shot changes in a video to extract the key frame of each shot for content-based retrieval and browsing. The QBIC system [6] supports querying-by-example on image and video databases. A logic-based language considering spatial and temporal descriptions was designed to support a visual query interface on image sequences [2]. [8] proposed a data model, in which the metadata of the media content were used to provide similarity browsing of multimedia data.

In our previous work, the problems of content-based indexing and querying capabilities for various types of media databases were studied. A content-based video query language (CVQL) [13] was proposed for querying video data. The relative position between symbols and the motion of symbols in a video can be specified by CVQL. Furthermore, an indexing strategy based on *chord* was proposed for querying music data by giving a piece of music [5].

* this work was partially supported by the Republic of China National Science Council under Contract No. NSC 86-2213-E-007-017.

[1] Dr. Koh is now an assistant professor in the Department of Information and Computer Eduation at National Taiwan Normal University.

The development of computer networks in recent years has made data sharing among databases important. We have proposed the strategies for schema integration and query processing in a multidatabase system which consists of object databases [10, 11, 12]. Assume a multidatabase system which consists of individually developed media databases and traditional *alphanumeric databases*. More powerful querying capabilities can be achieved by integrating the information in the media databases and alphanumeric databases. For example, the users may want to retrieve the movies whose directors have ever won the Oscar award. Since the predicative condition cannot be derived from the video content directly, the query cannot be processed in a video database. However, it is possible that an existing alphanumeric database contains the personal information of the directors. In such a environment, two kinds of *semantic relationships* among objects may exist. The first one is the *identity relationship* which implies that the involved objects represent the same real-world entity. These objects are named *identical objects*. Another kind of relationship is the *composition relationship* which denotes that the data for representing one object are included in the content of another object. In this paper, we present a system named IMADS(Integrated Media and Alphanumeric Database System) which is being developed to achieve a more powerful querying capability and a higher degree of information sharing. A query language IMAQL is designed for this purpose. The two kinds of semantic relationships are used to integrate the related information in the multidatabase system. The identical objects are integrated to an *entity* in IMADS, which represents a real-world entity. The queries in IMAQL are specified based on the entities. The desired entities can be retrieved by giving predicates on the related entities of different types through the composition relationships.

The work on multimedia databases mainly focuses on designing a database to manage media and alphanumeric data. In order to manage media data in an alphanumeric database [16], a *pseudo attribute* was proposed to denote the features of the media data in the alphanumeric database. The users can therefore specify restrictions on media data by giving predicates on the pseudo attributes in an alphanumeric form. [15] designed a content-based retrieval engine for multimedia data, where the similarity, fuzzy, and text retrieval functions were provided. However, in these approaches, only information in single media data can be used to specify query predicates.

In our approach, we use the concept of entity and the relationships among entities to specify query predicates in a multidatabase system consisting of multiple media and alphanumeric databases. [9] proposed a multimedia query language for retrieving multimedia documents which consist of multiple media segments. The users can specify the content of each segment as well as spatial and temporal predicates among multiple media segments. However, the entity-level information was not considered in the content of each segment. A data modeling strategy called *media abstraction* was illustrated in [3] to provide a unified interface for integrating media data of different types. Although a language was proposed, the language was logic-based which was difficult for the users to use.

This paper is organized as follows. The next section clarifies the basic as-

sumptions of the integrated environment. Various kinds of semantic relationships among objects in media and alphanumeric databases which contribute to the enhanced query capabilities are also introduced. Section 3 presents the syntax of the query language IMAQL. A graphic query interface for helping users to query the integrated system is shown in Section 4. Finally, section 5 concludes this paper with a discussion of the future works.

2 IMADS and the Semantic Relationships

2.1 The Integrated Environment

Assume IMADS consists of various types of media databases and an object-oriented database with alphanumeric data. Let a *media object* denote a data item in a media database, such as a movie in a video database. The symbols, voice, or words which can be extracted from a media object are named *symbol objects*. For example, a car appearing in a movie is a symbol object. In addition, a data item in an alphanumeric database is named an *alphanumeric object*. All the involved media databases in the system provide the content-based querying capabilities. That is, the symbol objects in a media object can be detected, and the indices on the features of the symbol objects are constructed. The classifications for categorizing the media and symbol objects respectively are also provided.

2.2 The Semantic Relationships

In IMADS, the *composition* or *identity* relationships may exist among media, symbol, or alphanumeric objects. Such kinds of relationships are called the *semantic relationships*, which are divided into the following five kinds.

1. The composition relationship between a media object and a symbol object: For example, the voice of a person appearing in an audio results in this kind of relationship between the audio and the voice of the person.
2. The composition relationship between two media objects: For example, a movie in a video database contains a song which is also represented as a media object in an audio database.
3. The identity relationship between two symbol objects: For example, a symbol object in an image and another in a video represent the same house.
4. The identity relationship between a media object and an alphanumeric object: For example, the director of a movie in a video database is represented by an object in an object-oriented database.
5. The identity relationship between a symbol object and an alphanumeric object: For example, John is represented as a symbol object in a video and also as an alphanumeric object in class **Person**.

The composition relationships can be added with descriptions in order to better represent the semantics of the relationships. For example, if a song is the

main title of a movie, "main title" represents the semantics of the composition relationship.

Suppose objects **o1** and **o2** have a semantic relationship. The information concerning **o2** can thus be used in the predicates for retrieving **o1**. Furthermore, the predicates for retrieving **o1** can involve the information of object **o3** if **o2** and **o3** also have a semantic relationship.

The real-world entities represented in IMADS are divided into three kinds: *media entities*, *symbol entities*, and *alphanumeric entities*. A media entity is represented by a media object in the media databases combined with its possible identical alphanumeric object. The media entities are further divided into video, image, audio, and document types. The symbol entities are the symbol objects together with their possible identical alphanumeric objects. Those alphanumeric objects which have neither identical media objects nor identical symbol objects belong to alphanumeric entities.

3 Query Language IMAQL

3.1 Basic Features of IMAQL

An IMAQL query mainly includes two parts:

Retrieve *Target-clause*;
From *R-P-clause*.

Target-clause The *Target-clause* specifies the result to be retrieved by the query, which has one or a list of notation v or $v.P$ separated by commas. v is a variable bound to entities in the *R-P-clause*, and P is a *path expression* proposed in object-oriented databases for denoting a property of the candidate entities.

R-P-clause The *R-P-clause* following the keyword **From** denotes the searching ranges and predicates of the query, which consists of two parts:

Range-clause
Predicate-clause.

The *Range-clause* defines the searching range for the following *Predicate-clause*. It has the following form: *entity-type*(*classification*)**:** e,
where the *entity-type* denotes the type of entities being evaluated. The *entity-type* can be one of the media types as specified by **Video**, **Image**, **Audio**, or **Document**. It also can be the symbol type denoted by **Symbol** or the alphanumeric type by **AlphaNum**. According to the specified entity type, the users can further specify a specific classification of the type to be the searching space. The entities in the specified range are called *candidate entities* of the *Predicate-clause*. e is a variable bound to the candidate entities.

The *Predicate-clause* specifies the restrictions on the associated candidate entities. It is defined as a boolean combination of predicates by using the logical connectives **and** and **or**. A predicate can be one of the following forms.

1. *containment predicate*
   A containment predicate restricts the candidate entities to those which contain a specific entity.

A *simple containment predicate* has the form: **contain** {*Component*}, where *Component* is denoted as: [*r-semantics*] *Sub-R-P-clause*. The *r-semantics* is used to specify the semantics of the composition relationship between a candidate entity and the component entity. The *Sub-R-P-clause* specifies the searching range and the predicates on the component entity.

A *complex containment predicate* has the following form:

contain {*Component 1, Component 2, ..., Component n*}
where { *S-T-predicates* },

where *Component i* ($1 \leq i \leq n$) has the same form as the *Component* in a simple containment predicate. A complex containment predicate restrains the candidate entities to those which contain a specific set of entities. In addition, the temporal and/or spatial relationships among the set of entities are specified in the *S-T-predicates*. The keyword **where** and the following *S-T-predicates* are omitted if no temporal or spatial predicates should be specified.

In this paper, three primitive functions are provided for denoting simple spatial and temporal predicates.

- **OMove(***v***)** *xy-expression*
  This function is used to specify predicates on the movement of a symbol entity. *v* is a variable bound to an entity. The *xy-expression* has the form: [**x** α 0, **y** α 0], where α is one of the comparators $<$, $=$, and $>$. It denotes the movement on the horizontal and vertical directions. For example, it represents that the entity moves to the right if $\mathbf{x} > 0$ is specified in the *xy-expression*. $\mathbf{y} < 0$ denotes that the entity moves down.
- **RP(***v1, v2***)** *xy-expression*
  The **RP** function specifies the relative position between two entities. *v1* and *v2* are variables bound to entities. The *xy-expression* has the same form as that in the **OMove** function except that it denotes the relative position from *v1* to *v2*. As an example, [$\mathbf{x} < 0$, $\mathbf{y} > 0$] denotes that *v1* is located on the upper-left side of *v2*.
- **RT(***v1, v2***)** *t-expression*
  The **RT** function specifies the predicate on the relatively appearing time of two entities. *t-expression* has the form: [**t** α 0]. α can be $<$, $=$, or $>$. If $\mathbf{t} < 0$, *v1* appears before *v2*. $\mathbf{t} > 0$ denotes that *v1* appears after *v2*. Otherwise, two entities appears at the same time.

The *S-T-predicates* is a sequence of these temporal and/or spatial predicates connected by logical connectives **and** and/or **or**.

2. *similarity predicate*
   A similarity predicate is used to denote the similarity retrieval by giving an example query, which is specified as: **similar-to(** *f-name* **)**,
   where *f-name* is the name of a file which stores a media object. The similarity degree between a candidate entity and the given media instance is computed by a similarity function. The predicate is evaluated to be *true* if the degree is less than a threshold value.
3. *appearing-in predicate*

An appearing-in predicate restrains the candidate entities to those appearing in a specific entity. A *simple appearing-in predicate* has the form:

appear-in {*Composition* }**.**

The *Composition* has the form: [*r-semantics*] *Super-R-P-clause*.

The *r-semantics* denotes the semantics of the relationship between the candidate entity and the specified composition. The searching range and the predicates on the composition are specified in the *Super-R-P-clause*.

4. *property predicate*

   A property predicate is used to specify predicates on the property of the candidate entity. It has the form: **with**(*P*):*p* {*P-clause*},
   where *P* is a path expression denoting a property of the candidate entities. *p* is a variable bound to the value of the property of a candidate entity one at a time, which can be omitted. The entities representing the values of the property become the candidate entities of the *P-clause*. If no predicate is specified on the property, {*P-clause*} is omitted.

```
Q1:
Retrieve M;
From Video(Movie): M
        contain{[main title]Audio(Music):
                similar-to(ex_music)}
```

```
Q2:
Retrieve P;
From Video(Movie):
        contain{Symbol(Person): X
                appear_in{Image(Picture): P},
                 Symbol(Car):C}
         where{RP(X,C)[x=0,y>0]
             and OMove(X)[x>0,y<0]
             and OMove(C)[x<0,y=0]
```

Fig. 1. The example queries

Another kind of predicates called *primitive predicates* are of the form: $\theta\ t$, where t is a constant, and θ is one of the *primitive comparators*: $<$, $\leq$, $=$, $\geq$, or $>$. The primitive predicates are used when the associated candidate entities belong to string or integer type.

In the examples shown in Figure 1, **Video(**Movie**)** is a searching range, followed by the predicates on the searching range. The containment predicate in **Q1** specifies that the interested movies contain a specific music. The [main title] in the predicate denotes that the music should be the main title of the movies. Similarly, the containment predicates in **Q2** restrict the symbols in the interested movies. The clause following the keyword **Where** specifies the relative positions and motions of these symbols in the movies. Since a media entity may consist of various components, the searching ranges for each component should be separately specified. For example, **Audio**(Music) in **Q1**, **Symbol**(Person) and **Symbol**(Car) in **Q2** are the searching ranges of the components. Furthermore, predicates can be specified on the components. Take **Q1** as an example, the contained music is restricted to be similar to a given example by the similarity predicate: **similar-to**(ex_music). In **Q2**, the contained symbol person is

```
Q3:
Retrieve Y.name;
From Video(Movie):
       with(title):{="die hard"}
  and contain{Symbol(Person): X
                 with(name):{="Bruce Willis}
            and with(height): h,
                 Symbol(Person): Y
                 with(height):{>h}}

Q4:
Retrievei X;
From Video(Movie): X
                 with(director): P
  and contain{Symbol(Person): P,
                 Symbol(Car): C}
        where{RP(P,C)[x=0,y≥0]
          and OMove(C)[x>0,y=0]
          and OMove(P)[x>0,y=0]}

Q5:
Retrieve n;
From Video(Movie):
       with(title):{="die hard"}
  and contain{Symbol(Man):
                 with(height): h}
  and contain{Symbol(Man):P
                 with(height):{ ≥ all h}
           and with(name):n}

Q6:
Retrieve n;
From Symbol(Person):
       with(name):n
  and appear_in{all Video(Movie):
 with(director.name){="Jackie Chen"}}

Q7:
Retrieve M;
From Video(Movie):M
       contain{Symbol(Person): P}
  and all P with(height):{<160}}

Q8:
Retrieve count(M);
From Video(Movie):M
        contain{Symbol(Male):P}
    and with(director):P

Q9:
Retrieve M;
From Video(Movie):M
       contain{Symbol(Person):P}
and Is-Class(P)group-by(M)="Female"}
```

Fig. 2. Example queries **Q3** to **Q9**

restrained to appear in certain picture. Therefore, an *R-P-clause* has a nested structure; that is, there can be another *R-P-clause* in a predicate. This nested structure also applies to the appearing-in and property predicates.

3.2 Extended Features

Join There are two ways for denoting the joins. The first way relates the entities by comparing their values. A join is also used to denote the identity of entities by assigning the same variable to bind the entities. For example, query **Q3** in Figure 2 retrieves the names of persons who appear in the movie "die hard" and are taller than the person named **Bruce Willis** also appearing in the movie. As another example shown in Figure 2, the movies whose directors also act in the movies by standing on top of a car running toward right are retrieved by **Q4**.

Quantifier The default quantifier for the predicates is the existential quantifier. The universal quantifier on the predicates should be specified explicitly by using the keyword **all**. The situations where the universal quantifier can be specified in a predicate are illustrated as the following.

- Applied in primitive predicates.
  For example, the example query **Q5** in Figure 2 shows how to retrieve the name of the tallest man in movie "Die Hard."
- Applied in containment/appearing-in predicates.
  For example, query **Q6** retrieves the names of persons who appear in all the

movies directed by Jackie Chen.

In another situation, the users may require that a specific set of entities all satisfy some restrictions. As the example **Q7** shows, it retrieves the movie in which all the persons are shorter than 160cm.

Aggregation Functions The usage of an aggregation function is as in the form: *Function-name*(u)**group-by**(v), where u and v are variables bound to entities. For each entity bound by v, the entities bound by u and satisfying the associated predicates are placed in a group for the computation of the aggregation function. The **group-by**(v) is an option, which can be omitted.

The aggregation functions can be specified in the *Target-clause* and also in the predicates. According to the type of entities, different aggregation functions are provided. The example query **Q8** retrieves the number of movies whose directors are male and also as actors in the movies. The **Is-Class** function returns the minimum common classification for the set of involved entities. For example, the movies which only have actresses can be retrieved by example query **Q9**.

4 Query Interface

A query interface is designed for providing a user-friendly interface to query the IMADS system. For simplifying the usage of the query interface, the extended features proposed by IMAQL have not been provided in the interface. Suppose we would like to retrieve the pictures of a person who appears in a movie. In addition, the movie contain two specifiable shots. In the beginning of the movie, a person jumps down from a car and the background includes some flowers on the right and a tree on the left. After a period of time, a subway appears and a song is playing at the same time. Furthermore, the song is sung by Madonna. The query is specified by the query interface as Figure 3(a) to 3(f).

The query interface is shown as in Figure 3(a) when beginning a new query. The users use a mouse to choose one of the querying domains in order to specify the type of entities being searched. Then a new window appears and the classification of the specified domain can be selected. In our example, the classification **Picture** of image entities is selected. The menu item labeled by **Predicate** is used to select the kind of predicates to be specified. The item **Contain** listed in the **Predicate** menu is selected and a *containment-predicate window* appears as in Figure 3(b). The contained entities and their spatial relationships in the searched picture are described in the containment-predicate window. As figure 3(b) shows, it denotes that a person appears in the specified position of the searched picture. At the same time, a window captioned by **Global View of Query** shows all the specified entities in the query as shown in Figure 3(b). More predicates can be specified on the entities contained in **Picture x** by selecting the associated items in **Global View of Query** window. In order to specify predicates on entities in which the person appears, the **Appear in** item in the **Predicate** memu is selected and an *appearing-in-predicate* window appears. We can specify that the person in the picture also appears in a movie as shown in

Figure 3(c). Similarly, the containment predicates can be specified on the movie as Figure 3(d). Since a movie is a video entity, the time issue should be considered in the predicates. The *relative time bar* is used for specifying the temporal relationships be tween the contained entities. The button on the time bar can be moved left or right to denote the relatively appearing time of the contained entities in a time sequence. In addition, the motion of a symbol can be specified by giving the direction and offset. As Figure 3(d) shows, it denotes that a person jumps down from a car and the background includes some flowers on the right and a tree on the left. By moving the button on the time bar right, another shot which appears after the previous shot in the movie is described by Figure 3(e). In the shot, a subway appears and a song is playing at the same time. In order to specify predicates on properties of the song, the **Property** item in the **Predicate** memu is selected and a *property-predicate* window shows the properties of a song. As Figure 3(f) shows, the name of the singer is specified to be Madonna.

We have implemented the query interface by the *Java* language [14] in the environment of Window95 operating system. The interface can be accessed and used via the internet browsers by linking to the following URL: http://piggy.cs.nthu.edu.tw/imadbs/imadbs.html. Efficient query processing strategies are currently under development. Integrating the video database and the music database, which we have independently implemented, in IMADS is also being conducted.

5 Conclusion

In this paper, a multidatabase system which consists of media databases and traditional alphanumeric databases is discussed. The two kinds of semantic relationships, identity and composition relationships, among objects are introduced for integrating the information in these databases to construct an integrated system IMADS. A query language IMAQL is proposed to provide powerful querying capabilities in IMADS. The identical objects in the multidatabase system, which denote the same real-world entity are integrated and represented by an entity in IMADS. The information of all the identical objects are considered to be the features of the associated entity. The queries in IMAQL are based on the entities. IMAQL provides the capabilities for specifying predicates on various types of media data and alphanumeric data. Moreover, the desired entity can be retrieved by giving the predicates on the related entities of different types through composition relationships and properties. The join operations, quantifiers, and aggregation functions are also proposed. Finally, a user-friendly query interface is presented which provides the facilities of incrementally querying IMADS and browsing the results.

It is difficult to automatically detect the semantic relationships. Developing a tool by combining the knowledge databases and the techniques of pattern recognition to semi-automatically detect these relationships is an important research issue, which is currently under our consideration.

References

1. Y.A. Aslandogan et al., Design, Implementation and Evaluation of SCORE (A System for COntent based REtrieval of pictures), *Proc. IEEE Intl. Conf. Data Engineering*, (1995) pp.280-287.
2. A.D. Bimbo, E. Vicario, and D. Zingoni, Symbolic Description and Visual Querying of Image Sequences Using Spatial-Temporal Logic, *IEEE Trans. Knowledge and Data Engineering*, 7(4) (1995) pp.609-621.
3. A. Brink, S. Marcus, and V.S. Subrahmanian, Heterogeneous Multimedia Reasoning, *IEEE Computer*, Sep. (1995) pp.33-39.
4. S.K. Chang and A. Hsu, Image Information Systems: Where Do We Go From Here?, *IEEE Trans. Knowledge and Data Engineering*, 4(5) (1992) pp.431-442.
5. T.C. Chou, A.L.P. Chen, and C.C. Liu, Music Databases: Indexing Techniques and Implementation, *Proc. IEEE International Workshop on Multi-Media Database Management Systems* (1996), pp.46-53.
6. M. Flickner et al., Query by Image and Video Content: The QBIC System, *IEEE Computer*, Sep. (1995) pp.49-56.
7. Gibbs, C. Breiteneder and D. Tsichritzis, Audio/Video Databases: An Object-Oriented Approach, *Proc. IEEE Intl. Conf. Data Engineering*, (1993) pp.381-390. *IEEE Multimedia*, 1 (1) (1994) pp.12-24.
8. W.I. Grosky, F. Fotouhi, and I.K. Sethi, Using Metadata for the Intelligent Browsing of Structured Media Objects, *ACM SIGMOD RECORD*, 23 (4) (1994) pp.49-56. *Proc. the 20th VLDB Conference*, (1994) pp.297-308.
9. N. Hirzalla and A. Karmouch, A Multimedia Query Specification Language, *Proc. International Workshop on Multimedia Database Management Systems*, (1995) pp.66-73.
10. J.L. Koh and A.L.P. Chen, Integration of Heterogeneous Object Schemas, *Lecture Notes in Computer Sciences : Entity-Relationship Approach-ER '93*, Vol. 823, Springer-Verlang: Berlin, (1993) pp.297-314.
11. J.L. Koh and A.L.P. Chen, A Mapping Strategy for Querying Multiple Object Databases with a Global Object Schema, *Proceedings of IEEE the fifth International Workshop on Research Issues in Data Engineering*, (1995) pp.50-57.
12. J.L. Koh and A.L.P. Chen, Query Execution Strategies for Missing Data in Distributed Heterogeneous Object Databases, *Proceedings of IEEE the 16th International Conference on Distributed Computing Systems*, (1996) pp.466-473.
13. T.C.K. Kuo and A.L.P. Chen, A Content-Based Query Language for Video Databases, *Proceedings of IEEE International Conference on Multimedia Computing and Systems* (1996), pp.209-214.
14. Sun Microsystems, Inc., JavaSoft, 1995.
15. J.K. Wu, A.D. Narasimhalu, B.M. Mehtre, C.P. Lam, Y.J. Gao, CORE: a Content-Based Retrieval Engine for Multimedia Information Systems, *ACM Multimedia Systems*, 3 (1) (1995) pp.25-41.
16. A. Yoshitaka, S. Kishida, M. Hirakawa, and T. Ichikawa, Knowledge-Assisted Content-Based Retrieval for Multimedia Databases, *IEEE Multimedia*, 1 (4) (1994) pp.12-21.
17. H.J. Zhang, C.Y. Low, S.W. Smoliar, and J.H. Wu, Video Parsing, Retrieval and Browsing: An Integrated and Content-Based Solution, *Proc. ACM Multimedia*, (1995) pp.15-24.

This article was processed using the LaTeX macro package with LLNCS style