(2)

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

Design and Implementation of
a Multimedia DBMS
: Retrieval Management

by

Pongsuwan, Wuttipong

September, 1990

Thesis Advisor: Vincent Y. Lum

Unclassified

Security Classification of this page

REPORT DOCUMENTATION PAGE

1a Report Security Classification Unclassified	1b Restrictive Markings
2a Security Classification Authority	3 Distribution Availability of Report Approved for public release; distribution is unlimited.
2b Declassification/Downgrading Schedule	
4 Performing Organization Report Number(s)	5 Monitoring Organization Report Number(s)
6a Name of Performing Organization Naval Postgraduate School 6b Office Symbol *(If Applicable)* 52	7a Name of Monitoring Organization Naval Postgraduate School
6c Address *(city, state, and ZIP code)* Monterey, CA 93943-5000	7b Address *(city, state, and ZIP code)* Monterey, CA 93943-5000
8a Name of Funding/Sponsoring Organization 8b Office Symbol *(If Applicable)*	9 Procurement Instrument Identification Number
8c Address *(city, state, and ZIP code)*	10 Source of Funding Numbers Program Element Number / Project No / Task No / Work Unit Accession No

11 Title *(Include Security Classification)*
Design and Implementation of a Multimedia DBMS: Retrieval Management (Unclassified)

12 Personal Author(s)
Pongsuwan, Wuttipong

13a Type of Report	13b Time Covered	14 Date of Report *(year, month,day)*	15 Page Count
Master's Thesis	From August 89 To September 90	September 1990	141

16 Supplementary Notation The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17 Cosati Codes			18 Subject Terms *(continue on reverse if necessary and identify by block number)*
Field	Group	Subgroup	Multimedia Database Management System, Multimedia, DBMS, MDBMS, Image Database.

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Current conventional Database Management Systems (DBMS) manage only alphanumeric data. However, data to be stored in the future is expected to include some multimedia form, such as images, graphics, sounds or signals. The structure and the semantics of the media data and the operations on that data are complex. It is not clear what requirements are needed in a DBMS to manage this kind of data. It is also not clear what is needed in the data model to support this kind of data; nor what the user interface should be for such a system. The goal of the Multimedia Database Management System project in the computer science department of the Naval Post Graduate School is to build into a Database Management System (DBMS) the capability to manage multimedia data, as well as the formatted data, and define operations on multimedia data. This thesis, focusing only on the media data of image and sound, first describes the operations of such a system, then discusses the general design of it, and finally outline the detailed design and implementation of the retrieval operation.

20 Distribution/Availability of Abstract	21 Abstract Security Classification
[X] unclassified/unlimited [ ] same as report [ ] DTIC users	Unclassified

22a Name of Responsible Individual	22b Telephone *(Include Area code)*	22c Office Symbol
Vincent Y. Lum	(408) 646-2693	52Lu

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted security classification of this page

All other editions are obsolete Unclassified

Approved for public release; distribution is unlimited.

DESIGN AND IMPLEMENTATION OF A MULTIMEDIA DBMS: RETRIEVAL MANAGEMENT

by

Wuttipong Pongsuwan
Lieutenant, Royal Thai Navy
B.S., Royal Thai Naval Academy, 1985

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 1990

Author: _____
Wuttipong Pongsuwan

Approved by: _____
Vincent Y. Lum, Thesis Advisor

_____
C. Thomas Wu, Second Reader

_____
Robert McGhee, Chairman,
Department of Computer Science

ABSTRACT

Current conventional Database Management Systems (DBMS) manage only alphanumeric data. However, data to be stored in the future is expected to include some multimedia form, such as images, graphics, sounds or signals. The structure and the semantics of the media data and the operations on that data are complex. It is not clear what requirements are needed in a DBMS to manage this kind of data. It is also not clear what is needed in the data model to support this kind of data; nor what the user interface should be for such a system. The goal of the Multimedia Database Management System project in the computer science department of the Naval Post Graduate School is to build into a Database Management System (DBMS) the capability to manage multimedia data, as well as the formatted data, and define operations on multimedia data. This thesis, focusing only on the media data of image and sound, first describes the operations of such a system, then discusses the general design of it, and finally outline the detailed design and implementation of the retrieval operation.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

I thank Dr. Vincent Y. Lum for all his support and encouragement throughout the development of MDBMS. He gave me guidance when I needed it . He is a great teacher and advisor.

Most important I thank my lovely wife, Rawiwan and my daughter Thanaporn, for their patience and support during the past two years at the Naval Post Graduate School.

I. INTRODUCTION

A. BACKGROUND

Current conventional Database Management Systems (DBMS) manage only formatted data (eg. alphanumeric data). Multimedia data such as image, sound, graphics and signals. are generally ignored. Many DBMS applications routinely need multimedia data or media data (these two terms will be used interchangeably in this thesis) as well as formatted data. Current technology allows us to keep these different types of media data in separate files. That is, a single image or a single signal, which we call a "media object" in this thesis, is an instance of media data and will occupy one distinct file. Although the term object is used, it is obvious that a media object is merely a single value of an instance, as in a normal database in which an instance of the value of the age attribute is 35. In multimedia data an image is an object but is also the value of the attribute picture. It does not require much imagination to see that under the circumstances, a user would soon lose track of the "objects", even for a very small application.

Handling data means allowing the storage and searching by the content of the data we process. Immediately, one would then ask the question of how to handle content search in multimedia data. There is no question that this is a difficult problem. One must find ways to handle a very large amount of multimedia data, with the capability to search and find the appropriate data conveniently and efficiently based on its contents.

A similar problem, though in much simpler form, was encountered earlier when dealing with the more "standard" types of alphanumeric data. Database management systems DBMS were developed as a result of users trying to solve the problem. The power of a DBMS is well recognized today and there is no need for us to discuss these systems here. What one wishes to develop is a technology that would allow us to handle

multimedia data as conveniently as we can process the standard data. The Multimedia Database System Project (MDBMS) in the Computer Science Department of the Naval Post Graduate School was formed for this purpose [WK87, LM88].

The need to support multimedia data processing is actually a natural extension of the use of the computer to process standard data. Applications generally need not only formatted data, but also the "unformatted" data or media data. Unformatted data, including image, video, sound and signal are usually stored as multimedia objects. For example, in a person database, one would store the various data such as an birthday, an address, a job title, about an individual. In many applications one would most likely want to store and process the photo of the individual as well, if the technology allows this to be done easily. In military applications, we need access to both formatted data and unformatted data in order to make decisions on many operations. In the past we processed these operations manually, and it required much effort to get through the formatted and unformatted data for pertinent information. Currently, and even more so in the future, computers are used to assist the users. Today storage and processing media data in routine applications are still not so simple, although are very close from the technology's standpoint.

Aside from the MDBMS project here, a number of projects have been established to do research in multimedia data processing. Among these include the following: the MINOS project at the University of Waterloo [Ch86], which is aimed at the management of documents containing multimedia data; the ORION system at MCC in Austin, that contains a Multimedia Information Manager (MIM) [WK87] for processing multimedia data; the IBM Tokyo Research Laboratory "mixed object database systems," MODES1 and MODES2 [KKS87]; and an ESPRIT project in Europe designing a multimedia filing system called MULTOS [Be85,Be86, BRG88]. A discussion of these projects is presented in [LM88] and [MLW89] and will not be repeated here. Today, MDBMS

research and development is still in the infancy stage. Even the definition of the functionality of a multimedia database management system (MDBMS) is still an open issue.

Recently the management of the multimedia data on the personal computer has grown rapidly. The hypertext and hypermedia data management in the Macintosh computer with a hypercard application has many users including the ARGOS project [WNTA89] being developed at Naval Post Graduate School. Hypertext uses the idea of card stacks in which each card in a stack contains many objects called buttons and fields corresponding to functions that can be invoked for processing by clicking the mouse set to that icon on the screen. A problem of using hypertext and hypermedia is that users cannot query the data as done in the conventional database systems. Users can easily get lost during the search process. Another problem is that hypertext uses the hierarchical database approach and its users have to go down the branches of the tree to get to the card needed. Additionally, hypertext uses an interpreter to process user commands. Both of these approaches are time-consuming. Further, although hypertext and hypermedia data management give the users much more power to process their data, it only works on the microcomputers in the single user environment. Data cannot be shared as in the normal central or distributed database systems. To avoid the above restrictions and shortcomings, a standard DBMS with the extended capability to process the media data was introduced [MLW89]. As discussed in Chapter II, this multimedia database manage system (MDBMS) consists of subsystems to manage conventional databases and advanced databases supporting media object management, and the integration layer that provides a uniform interface to access alphanumeric data, media data, or mixture of both.

B. SAMPLE APPLICATIONS

In the last section we discussed the background related to multimedia data processing. In this section we shall discuss in substantial detail the scenarios of some applications. The

purpose is to give the readers a better understanding in the design and operation of our system for multimedia data processing.

Suppose, for example, in response to an occasion the chief of staff of the Navy wants to assign ships to an operation, and release information to the news media. To accomplish his job, he might need not only the formatted data which can be supported by the conventional database but also the unformatted (image, sound or signal) data in the database as well.

He might want to know which ships can be sent to that geographical area. In order to find which ships can get to that area, he would need to know the location of the ships, the travel time for the ships to get to the operation location, the firing power of the ship, and the personnel of the ships, such as the key officers. This kind of operation, however, only needs the formatted data.

Sometimes this person might want to see, in addition to the formatted data, the images of the ships and the pictures of the officers. For example he might want to release some pictures of certain ships and their officers to the news media. This kind of operation requires both formatted and multimedia data from the database.

In another occasion, the chief might want to release to the news media some photos in the Navy's collection that show the heroic efforts of the navy personnel, and their ships, rescuing civilians in a disaster at sea. The specific photos suitable for such an illustration are not known but all appropriate photos would be scanned to search for the desired ones. Such kind of search can only happen if the contents of the photos, a specific kind of media data, could be searched. Naturally one might also want other formatted data like date and place and time the photo have been taken, etc. to go with the media data.

In short it is safe to assume that both formatted data and media data may be needed in certain applications. Moreover, sometimes, not only we need to store and retrieve media data but also to search this kind of data based on their contents.

C. THE SCOPE OF THE THESIS

The general design of the overall application for multimedia database management system includes the design of the high level operations like table creation and data insertion, retrieval, update, and deletion, composing the main functions of any DBMS. Three students doing related work on their MS theses in the Computer Science Department of the Naval Post Graduate School are involved. The detail design and implementation for the creation of table and the insertion of tuples are given in the thesis by Pei [PE90]. The storage and management of sound data using an IBM compatible computer connected to the main database system is given in the thesis by Atila [AT90]. This thesis currently being read, will focus on the design and implementation of the retrieval operation. The design of the new system catalog for the new data type of the media of image and sound for retrieval as well as the other functions will be given. This thesis will show that, in order to retrieve the data, we have to first build temporary database tables for further processing.

Chapter II in this thesis will discuss the previous works in the Multimedia Database System Project including the architecture of the MDBMS system. Chapter III section A will discuss the environment in which the MDBMS is to be built. In section B of chapter III, the design of the MDBMS catalog will be described. In section C of the same chapter an overview of the database operations will be presented and in section D a detail description of the design of the retrieval operation will be given. Chapter IV concentrates on the implementation aspect of the retrieval operation. In section A of chapter IV, the interface design for the retrieval operation will be described; in section B of this chapter query processing for retrieval will be given; in section C of the same chapter the data

structures for the retrieval operation will be described; in section D the program structures for the retrieval operation will be presented; and in section E, a method to link and run the MDBMS will be given. Chapter V will present the conclusions and summary.

II. SURVEY OF PREVIOUS WORK

The work in the Multimedia Database System Project at the Computer Science Department of the Naval Post Graduate School began in 1988 [LM88, MLW89]. The first work was to design the architecture of the MDBMS to process multimedia data as conveniently as the processing of the standard data (formatted data). The gross architecture was composed of three parts. The first part is the MDBMS interface which is the interface between the users and the formatted data and media data. The second part is the standard DBMS which manages all the formatted data. The third is the Media Manager for media data. One may consider that the Media Manager to be composed of different subsystems which currently includes the Image Manager and the Sound Manager since only image and sound are supported. The detail discussion is in [LM88]. The architecture is as shown in Figure 1.

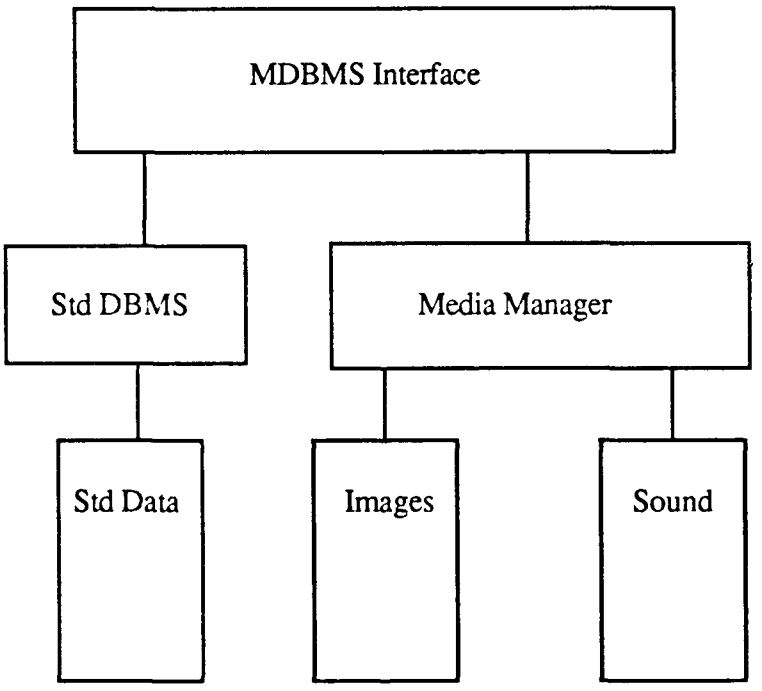

Figure 1. The architecture of the MDBMS

Conventional DBMS systems do not support multimedia data. To support multimedia data we must fit it into a data model. It has been determined that the abstract data type (ADT) concept is most appropriate for the task [LM88].

In the work of [LM88], it was proposed that media data types like image, sound, graphics, text or signal be defined and their operations constructed. When a media data type is encountered, the system will be able to support it through this new structure. For example, suppose the image data type is set up and can be used as an attribute domain. A relation PERSON (name, age, photo) can be defined where name and age are the normal data types (e.g. character and integer), and photo is of image type. The operations on such a data type, image, is given in [LM88].

Operationally, as stated in the previous chapter, the processing of media data type sometimes requires the recognition of the contents of the media data. Since automatic recognition of media contents by the computer is beyond the state of art, a proposal to supplement the raw media data with the descriptions in natural language form has been suggested.

Further, multimedia data are always accompanied by some standard formatted data called registration data. For images it could be resolution, pixel depth, source, date of capture, and colormap. The important issue of the registration data is that they are required if anything is to be done with the multimedia data at all, either to interpret them for replay or display, or to identify them and distinguish them from others. Registration data can easily be stored in the attributes and tuples of standard relational database systems, thus making the full power of query languages available to retrieve and manipulate them.

While registration data is indispensable, the description data, either formatted or unformatted for describing the contents of multimedia data generally are not. This

description data is redundant, because it generally repeats information already present in the image, text, or sound. However, because of the complexity and the depth of its information content, there is hardly any chance to perform efficiently a contents-oriented search on the media raw data themselves. Such kind of work requires much too much intelligence in a system than we know how to provide today. Thus, it is much easier and more effective to let a human user provide the description, just as an author provides abstract and keywords with an article. In either case the database should hold the result of the extraction, i.e. the description, and link it to the multimedia data. Thus, we have each instance of a multimedia data represented in three parts: registration data, raw data and description data as shown in Figure 2 [MLW88].

Multimedia data, their registrations and their descriptions can be used in various ways. Any access to the raw data must go "through" the registration data to make sure that the raw data are interpreted correctly. Editing operations on the raw data including filtering, clipping, bitmap operations for images, stripping of layout commands and control characters for text, etc. are permitted. Special operators that are applied to the media data can be distance and volume calculations on geometric data, or the addition of synonyms in the case of keywords. Sometimes, these operators can actually do a lot of processing without ever touching the raw data. Most of these operators cannot be implemented with only the commands of the query language. They need the features of a general-purpose programming language. New data models must allow them to be incorporated into the database as "procedure" or "method".

The media data type, which was previous defined in [MLW88] is IMAGE datatype. IMAGE is regarded as an abstract data type with its own set of operators or functions. By this design the IMAGE composed of three parts namely raw data (consist of a matrix of pixels), registration data (size of the image, resolution, encoding, colormap), and description data (description of the image). The operations of the relational database cannot

be performed directly on the data type IMAGE. They treat an IMAGE value as a whole, i.e. projection either drops it completely or keeps it in the result. The comparisons needed in selections and joins cannot be performed on the whole image. A detail discussion is in [MLW88].

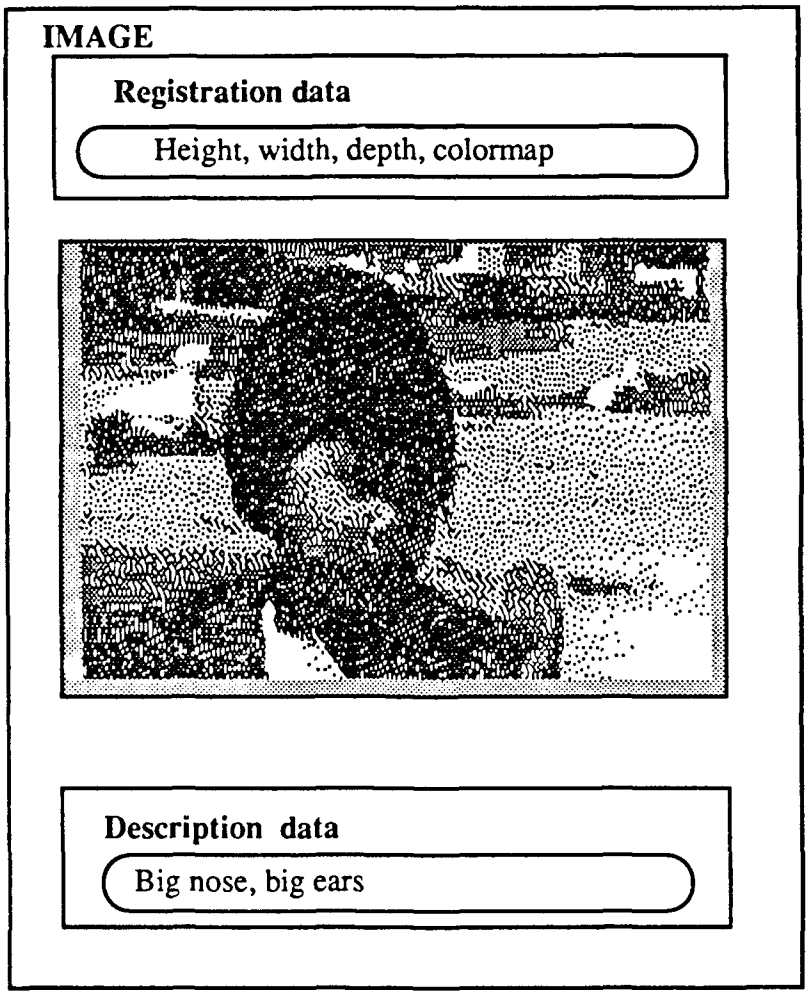

Figure 2 - Conceptual View of an Instance or Value of the Abstract Data Type IMAGE

The subcomponent Image Manager of the Media Manager, as shown in Figure 1, was implemented by Thomas [Th88]. Thomas provided the lower level applications for

processing image database which have the ability to search and store images in the DBMS. She provided the internal functions to process the image data as well. Thomas used a relational database that incorporated the IMAGE data type. This IMAGE data type can be modified to accommodate changes in the database environment with no modification from the user side of the database interface.

The subcomponent Sound Manager of the Media Manager as shown in Figure 1 was implemented by Sawyer [Sa88]. Sawyer provided a similar processing capability for the incorporation of sound data as done by Thomas to handle image data. At that time, SUN 3 Workstation did not provide support for sound data so an IBM-compatible PC was used for sound processing.

The content-description search for the media data was first implemented by Meyer-Wegener [LM90] by using a parser to parse the natural language descriptions and get the result in a form acceptable by Prolog. To parse natural language descriptions, one needs a dictionary to define the vocabulary. A dictionary had been built for this purpose. A detail description of the parser is given in Dulle [Du90].

The MDBMS prototype previously implemented was named DEMOS1. DEMOS1 was designed for managing only the image data types. DEMOS1 contains 2 parts: one is the Prolog portion for providing support for the natural language search; the second is an INGRES DBMS. The INGRES DBMS portion handles only media IMAGE relations. This prototype does not have an interface processing both formatted and media data. It can retrieve images via the identifiers of the images or their natural language descriptions.

The current prototype is an attempt to broaden the database handling capability by providing the integrated support of both formatted and media data. Its design and implementation, as described in chapter 3 and chapter 4 of this thesis, is based on the same architecture of the previous work. The new system will provide the high level operations

of table creation and data insertion, retrieval, deletion, and update for both formatted and media data.

III. DESIGN OF THE SYSTEM

A. SYSTEM ENVIRONMENT

As mentioned in the previous chapter, the prototyping effort for building this multimedia database system (MDBMS) began about two years ago [Th88, Sa88, MLW89]. For various reasons, INGRES was chosen to be the DBMS to manage the formatted data and the SUN workstations and servers with the UNIX operating system were chosen to be the system in which the multimedia database management system (MDBMS) was to be constructed. Because the SUN workstations in 1988 did not support sound, an IBM compatible PC was used to store sound data.

A number of restrictions are the consequence of using the INGRES DBMS. First, the INGRES version in which the original MDBMS prototype was constructed does not support user-defined abstract data types. Second, INGRES allows a maximum of 500 characters to be stored for a given attribute. Third, it does not allow its users to get the catalog information readily. Fourth, an intermediate interface below the language of SQL is not available for the INGRES users. That is, although SQL is supported on INGRES, SQL is compiled directly into INGRES low level code for execution. Each of the above restrictions affected the design and implementation of our MDBMS.

Although more recent versions of INGRES have removed some of the restrictions, significant recoding effort is required to make use of the new versions. As the prototype construction at this time is not intended to be a production system, but only as a demonstration of the various concepts, a decision was made not to recode. Similar situation occurs in the SUN workstations. New SUN workstations now support sound. But that would require substantial investment to purchase new hardware and recode some programs. It was decided that, instead of these investments, which would produce little

gain, the PC system would be retained to manage sound data and would be integrated into the MDBMS prototype as a backend server for sound by connecting it to the SUN system via a local area network, which in this case is the Ethernet.

The environment just discussed influences the design and implementation of the system and will be reflected in the various parts in this thesis as well as the theses by Pei [PE90] and Atila [AT90].

B. CATALOG DESIGN

Because the INGRES catalog management cannot accommodate the needs of the MDBMS, it becomes necessary to design and manage the MDBMS catalog ourselves. While it seemed advantageous in the beginning to use INGRES to manage our catalog tables, further investigation revealed that such an approach only complicates the MDBMS operations and produces no benefits. This occurs because INGRES does not know what MDBMS wants to do with the catalog information. With or without INGRES, MDBMS must manage its catalog as tables outside the INGRES system[PE90]. Using INGRES to manage the MDBMS catalog tables therefore would mean the addition of a layer and operations unnecessarily. The decision was made to create the catalog in the form of system tables in the internal memory throughout the operation of MDBMS. When a user signs off, the updated system tables are written out as files. When the user restarts the MDBMS, the files are read into memory again before any user operation is performed.

In our design, we have four tables or arrays to be used for storing our catalog information. The first is Table_List array which will contain the integer number that points to the entries in Table_Array. The reason why we have such a structure and the detailed use of this structure will not be given here but is given in [PE90]. Basically it is done for database maintenance purposes. The second table is Table_Array which is composed of table_name, table_key, att_count and att_entry: table_key denotes the number that will be

assigned to the media relation in the create table, att_count denotes the number of attributes in the table, and att_entry shows the starting point of the table in the att_array. The third table is the Media_Array table which keeps track of the media data that exist in our database. The fourth table is the Att_Array table which contains att_name, data_type next_index and value_entry. Att_name is the name of the attributes for any table; data_type is the data type for each attribute including formatted and media data type; next_index is the pointer to the next attribute in the sequence of attribute entries for a given table; and value_entry is the pointer to the array Value_Array, that has the attribute value for that attribute entry in the Att_Array. Actually, the Value_Array table has four arrays representing four data types which correspond to the value_entry in Att_Array. The four arrays are char, integer, real, and media data (image and sound) and these represent all the data types that the MDBMS supports. The Value_Array tables are used to store data before the data is input to the database and also before it is displayed for the user. The structures of the catalog tables discussed above are shown in Figure 3 through Figure 5. The structure of the Value_Array discussed above is shown in Figure 6.

Table_List :

1
2

Table_Array :

table_name	table_key	att_count	att_entry
EMPLOYEE	1	6	1
DEPT	2	4	7

Figure 3 Table_List and Table_Array tables for the catalog

Media_Array :

photo
voice
dphoto

Figure 4 Media_Array table for the catalog

Att_Array :

att_name	data_type	next_index	value_entry
fname	c20	2	
lname	c20	3	
salary	float	4	
dnum	integer	5	
›to	image	6	
∨oice	sound	-1	
dno	integer	8	
dname	c20	9	
dloc	c20	10	
dphoto	image	-1	

Figure 5 Att_Array table for the catalog

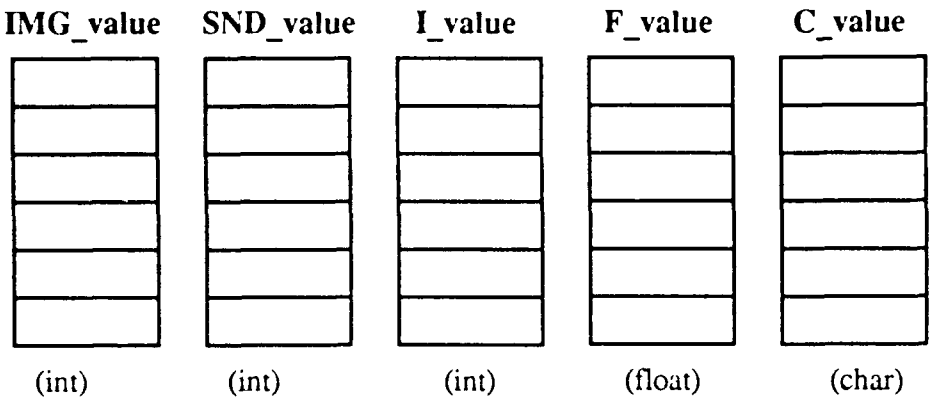

Figure 6 Value_Array tables

Let us illustrate the use of the catalog tables by means of an example. A user creates two relations named EMPLOYEE and DEPT. The EMPLOYEE relation has six attributes which are fname (c20), lname (c20), salary (float), dnum (integer), photo (image), voice (sound). The DEPT relation has four attributes which are dno (integer), dname (c20), dloc (c20), dphoto (image). First, when a relation is to be created, the catalog management part of the MDBMS will search through the list of relations, checking if a duplication exists for the new relation name with the previous relation names in Table_Array. If a duplication is found, the user has to reenter a different, new name. If no duplication exist, the system will put the new relation into the next slot of Table_Array and assign the value to the Table_List corresponding to that position. For example, a user enters a new relation name DEPT into the system. The system will check for duplication of relation names and if no duplicates exists, the system will insert DEPT into Table_Array at the next slot which is row 2. In the next step the system will insert the index number 2 into next slot of Table_List (as shown in Figure 3). The next step is to enter the first attribute name of the relation and its data type. This information is entered into the next slot in the Att_Array table. The position (or the row number) in Att_Array is then inserted into the Table_Array table in the field att_entry for the corresponding relation. For example, a user enters dno which is the first attribute for the relation DEPT. The system will search for the next available slot in table Att_Array, index 7 is the first available slot. The system then inserts

dno and its data type (e.g. integer in this case) into the Att_Array table and puts the index (e.g. 7) for that attribute in Att_Array (e.g. dno) into the att_entry column for relation DEPT (as shown in Figure 3 and Figure 5). When the user enters the next attribute in the relation (e.g. dname in DEPT) into Att_Array, the corresponding index must be put into the next_index column of the previous attribute (e.g. dno).

This goes on until no more attribute is to be inserted for the relation. Thus in our example, the user enters attribute(s) dname (c20), dloc (c20), dphoto (image) into the relation DEPT after dno. The attribute dname will be assigned to the next slot after dno, which is row 8, 8 is the value for next_index in the row where dno is located. The same operation will occur for dloc and dphoto. The next_index entry for last attribute will contain the end mark (-1). Since the dphoto is the last attribute for the DEPT relation, the next_index for the attribute dphoto therefore is -1 as shown in Figure 5. After finishing the creation of a relation the user can modify the relation by changing relation name, attribute names, the data types or deleting attributes. This kind of operation can be easily handled by adjusting the entries in the tables and the indexes of the array.

The system catalog discussed above will be used by all the operations in the MDBMS. The use of array index, compared to the use of pointer linked list structure, is judged to be superior; it saves a lot of time in searching the catalog tables and simplifies the implementation as well. However, while attributes are required to be unique within a user relation, same attribute names are permitted in different relations. While this situation works fine for formatted data because INGRES manages this kind of data and confusion will not arise, it creates problems for handling media data. In the MDBMS prototype, a separate relation, referred to as media relation in INGRES, is created for each media attribute. The name of this media relation is the same as the name of the media attribute. For example, a media relation is created for dphoto in DEPT. To avoid confusion and keep the media relations distinct when the same attribute name is used in more than one user

relations, the names of these media relations are appended by suffixes corresponding to the unique system identifiers assigned to the user relations. A detailed description about this will be given in the next section.

C. DATABASE OPERATIONS

The operations of the multimedia database management system on a high level will be the same as in a normal DBMS for formatted data, namely, creation of table and data insertion, retrieval, update and deletion. I shall discuss each one briefly and show how these operations are to be accomplished in the MDBMS prototype.

The creation of a relation in a database requires the structure of the relation, namely the attributes and attribute data types. The relation name, the attribute names in sequence and the attribute data types (which can be Integer, Real orChar) followed by the length information are needed. If all the attributes are the standard formatted data type, this information and the create table command can be passed directly to INGRES via the create table command of SQL. Because the media data type cannot be handled by the INGRES system, we must handle media data differently than the formatted data.

To solve the above problem, we do the following: When a media data type attribute is encountered, we enter into our catalog, specifically the Att_Array table in Figure 5, the appropriate media data type. We then request INGRES to create a media relation specifically for this media attribute. The structure of the media relation depends on the type of the media, which at this time can only be image and sound. Figure 8 shows the media relation for image data type and Figure 9 shows the media relation for the sound data type. The meanings of the attributes in media relation Photo (Figure 8) are as follows: i_id is the system assigned internal identifier that is used as the value to be entered in the attribute Photo in the user relation EMPLOYEE, f_id is the file name of the image and the exact path where this file exists, descrp is the natural language description of the content of the image

file, and height, width and depth are the parameters that are needed to display the image file. Similarly, for sound data type as in Figure 9, s_id and f_id play exactly the roles as in image data type, descrp is the natural language description of the content of the sound file, and freq, sample (for sampling rate), res (for resolution of the sound) and encoding are parameters that are needed to reproduce the sound recorded.

How to use these relations to handle media type can now be explained easily. The media data in a user relation will contain the appropriate id's corresponding to the media file names as given in the proper media table. Using the file names, the system can retrieve the media files accordingly. Thus, whenever an attribute is encountered, the system will first check to see if it is a media type. If not, nothing outside of INGRES management is needed. If media data is encountered, the system will separate the processing into two parts: one part consisting of the processing of the formatted data to be done by INGRES and the other part the processing of the media data done outside of INGRES using the information in the media relation.

Relation name EMPLOYEE

fname	lname	salary	dnum	photo	voice

Figure 7 User relation EMPLOYEE

Relation name PHOTO1

i_id	f_id	descrp	height	width	depth

Figure 8 media relation for attribute EMPLOYEE.photo

Relation name VOICE1

s_id	f_id	descrp	freq	samp	res	encoding
...	...	...	...	...	...	...

Figure 9 media relation for attribute EMPLOYEE.voice

Let us now illustrate by an example what has been just said. For example, if we want to create the relation EMPLOYEE, which has the following attributes: f_name (c20), l_name (c20), salary (float), photo (image), voice (sound), we cannot create the relation in the database directly using INGRES. We have to separate the creation into three parts to set up the three relations, one for the relation EMPLOYEE, one image media relation and one sound media relation, as given below:

1. EXEC SQL CREATE TABLE EMPLOYEE (*fname c20, lname c20, salary float,* dnum integer, photo integer, voice integer).

2. EXEC SQL CREATE TABLE PHOTO1 (i_id integer, f_id c64, descrp vchar500, height integer, width integer, depth integer).

3. EXEC SQL CREATE TABLE VOICE1 (s_id integer, f_id c64, descrp vchar500, freq float, samp float, res integer, encoding integer).

Figure 7 through Figure 9 show the final result of the three tables in the database. Note that the media tables have suffixes on it. As explained before, this is done because the names of the attributes are not required to be unique in the whole database although they are unique in the same relation. The suffixes to be added to the media attribute names are the table_key values in Table_Array for the corresponding relations. For example the image attribute photo will get the suffix 1 because the table_key value in the Table_Array

for the EMPLOYEE relation (first row in Figure 3) is 1. A detailed discussion of creation is given in [PE90].

The insertion of formatted data into the database can be easily accomplished by INGRES. The insertion of media data cannot be done directly. The insertion of media data into the database requires the establishment of media files in the system beforehand so that the system can insert the f_id into the media table. Moreover, as the media table have other attributes, like height, width, and depth for the image data and freq, samp, res, and encoding for the sound data, the values for these attributes must be derived first prior to the insertion operation.

For example, a user wants to put the image of an employee into the photo attribute of the EMPLOYEE relation, This person must digitize the image in the correct format for the system to display when requested. In our program, the image can be captured from a camcorder and digitized into a GIF (Graphic interchange format) file on a PC, This digitized image file is then transferred to the SUN system in which the MDBMS is implemented. After the GIF file has been transferred to the SUN system, we change the GIF file into the . N Raster format which can be displayed by using Sunview. A detail of the digitizing technique and the transformation of the file format will be provided in Appendix A.

When the user wants to put the image into the EMPLOYEE relation, he only inputs the file name of the image in the photo attribute of EMPLOYEE relation, the MDBMS will generate i_id and f_id and insert i_id into the photo attribute in the EMPLOYEE relation as well as i_id in the photo media relation namely PHOTO1. At the same time a procedure also extracts the data, height, width and depth to be inserted into the PHOTO1 relation. Similar operation occurs for the insertion of sound data. A detailed discussion of the insertion of image media data is given in [PE90] and sound data in [AT90]. The following

relations in Figure 10 through Figure 12 illustrate the case where user data has been inserted into the EMPLOYEE relation.

The input data in the attribute photo and voice in the EMPLOYEE relation are integer type and serve as the indexes to the i_id of the PHOTO1 relation and s_id of VOICE1 relation respectively. For example, ralph w. has photo in the first entry of relation PHOTO1 (which contains 'big nose'). When the user wants to input the value for attribute photo, he must input the image filename which is '/n/virgo/work/mdbms/p1.pix' into the system. The system will insert that filename into field f_id of the relation PHOTO1 which represents the media attribute photo. At the same time height, width and depth will be extracted out of the header file and inserted into the relation PHOTO1 as well. The system will assign i_id to that photo attribute (which is 1 in this example) and enter it in the photo attribute column in the relation EMPLOYEE and the i_id column of the relation PHOTO1. After he enters the filename into the database, the system will ask for the description. The user will insert the description of that photo which is 'big nose', the system will insert the description into attribute descrp column of the relation PHOTO1. The same operation will occur when he inserts voice into the tuple of ralph w. of the relation EMPLOYEE. He must input the sound filename which is '/n/virgo/work/mdbms /p1.snd' into the system. The system will insert that filename into f_id of relation VOICE1 which represents the media attribute voice. At the same time freq, samp, res and encoding will be extracted out of that file and inserted into the relation VOICE1. After that, the description of the voice will be inserted into the attribute descrp in the relation VOICE1 as it does for the photo (as shown in Figures 10 - 12).

Relation name EMPLOYEE

fname	lname	salary	dnum	photo	voice
ralph w.	west	80000	1	1	1
mark	hendrickson	25000	1	2	2
jim	huskins	30000	3	3	3
john	dulle	26000	2	4	4
...	...	...	...	...	...
george	wilson	32000	1	10	7

(The input data in the attribute photo and voice in the EMPLOYEE relation are integer type and are used as the indexes to the i_id of the PHOTO1 relation and s_id of the VOICE1 relation respectively.)

Figure 10 User relation EMPLOYEE after insertion

Relation name PHOTO1

i_id	f_id	descrp	height	width	depth
1	/n/virgo/mdbms/p1.pix	big nose	640	480	8
2	/n/virgo/mdbms/p2.pix	small eyes	640	480	8
3	/n/virgo/mdbms/p3.pix	big head	640	480	8
4	/n/virgo/mdbms/p4.pix	big eyes	640	480	8
...	...	...	...	...	...
10	/n/virgo/mdbms/p5.pix	big ears	640	480	8
...	...	...	...	...	...

Figure 11 media relation for attribute EMPLOYEE.photo after insertion

Relation name VOICE1

s_id	f_id	descrp	freq	samp	res	encoding
1	/n/virgo/mdbms/p1.snd	slow voice	10	10	10	4
2	/n/virgo/mdbms/p2.snd	ugly voice	10	10	10	4
3	/n/virgo/mdbms/p3.snd	meeting in Iraq	10	10	10	4
4	/n/virgo/mdbms/p5.snd	blast off	10	10	10	4
...	...	...	...	...	...	...
7	/n/virgo/mdbms/p9.snd	high pitch	10	10	10	4
...	...	...	...	...	...	...

Figure 12 media relation for attribute EMPLOYEE.voice after insertion

The update and delete operations are planned to work similarly as in the operations just described. However, since these operations are not being implemented at this time, they will not be discussed, as their discussion does not add to our understanding.

Retrieval of multimedia data is the emphasis area of this thesis. Its design and implementation will compose the rest of this thesis. Discussion of the design issues are presented in the next section of this chapter and the implementation aspects in the next chapter.

D. RETRIEVAL DESIGN

The design of the retrieval operation is the most complex of the various operations. As stated earlier, the MDBMS prototype will support the extended SQL or the equivalent operations. However, SQL on INGRES does not support user-defined media types; we have to achieve our goal by extending the SQL language and building more procedures to support the extended language [MLV89, LM90]. With the design to support media data as

described in the previous sections, an extended SQL query may have to be decomposed into multiple subqueries each of which must be individually processed, and the intermediate results of which must be recomposed to form the final result to be given to the user. The idea of query decomposition is as follows: If a query references only the formatted data, it can be passed directly to INGRES without modification; a decomposition is necessary whenever media data is referenced. The following query examples will illustrate the points and help us understand the operation.

1. SELECT fname, lname, salary

   FROM EMPLOYEE

   WHERE salary > 30000;

2. SELECT lname, photo

   FROM EMPLOYEE

   WHERE salary > 25000;

3. SELECT lname, salary, photo

   FROM EMPLOYEE

   WHERE CONTAIN (photo, 'big nose') ;

(Note that CONTAIN is a procedure which searches the photo descriptions for "big nose". This method of searching the contents of the multimedia data was previously presented in [LM89].)

4. SELECT lname, salary, photo

   FROM EMPLOYEE

   WHERE salary > 30000 and dnum = 1 and CONTAIN (photo, 'big nose') ;

5. SELECT EMPLOYEE.lname, EMPLOYEE.salary, department.dname

   FROM EMPLOYEE, DEPT

   WHERE EMPLOYEE.dnum = DEPT.dno;

6. SELECT EMPLOYEE.lname, EMPLOYEE.salary, department.dname

FROM EMPLOYEE, DEPT

WHERE EMPLOYEE.salary > 20000 and

CONTAIN (EMPLOYEE.photo, 'big nose') and

EMPLOYEE.dnum = DEPT.dno;

7. SELECT lname, fname, photo

FROM EMPLOYEE

WHERE CONTAIN (photo, "big nose") and

CONTAIN (voice, "blast off");

8. SELECT EMPLOYEE.lname, EMPLOYEE.fname, DEPT.dname

FROM EMPLOYEE, DEPT

WHERE CONTAIN (EMPLOYEE.photo, "big nose") and

CONTAIN (DEPT.dphoto, "pyramid style") and

(EMPLOYEE.dno = DEPT.dno);

Let us examine each query to see what is to be done:

Query 1: No media data is referenced we can just pass the query to the INGRES.

Query 2: The same as query 1 but the selection has one attribute with type image. This query will do the same as query 1 but the display process has an additional operation to display the image. It invokes a procedure which takes an i_id from the query result and find the tuple in the corresponding image media relation (eg. PHOTO1) whose i_id entry equals to the i_id of the query result.

Query 3: Media data is referenced by the description "big nose" so query decomposition is necessary. The query will divide the work into three subqueries as follows:

1. CREATE TABLE F1 AS

SELECT all

FROM EMPLOYEE;

2. CREATE TABLE M1 AS

SELECT i_id

FROM PHOTO1

WHERE CONTAIN (PHOTO1, 'big nose');

3. CREATE TABLE RESULT AS

SELECT fname,lname,salary

FROM F1

WHERE (F1.photo = M1.i_id);

Query 4: Is similar to query 3 but subquery 1 must be modified to become

1. CREATE TABLE F1 AS

SELECT all

FROM EMPLOYEE

WHERE salary > 30000;

The rest is as query 3.

Query 5: No media data is referenced. Pass the query directly to INGRES.

Query 6: The media data is referenced by the description "big nose" and decomposition is needed. The query is similar to query 3 and is broken into three subqueries as follows:

1. CREATE TABLE F1 AS

SELECT all

FROM EMPLOYEE, DEPT

WHERE EMPLOYEE.salary > 20000 and

EMPLOYEE.dnum = DEPT.dno;

2. CREATE TABLE M1 AS

SELECT i_id

FROM PHOTO1

WHERE CONTAIN (PHOTO1, 'big nose');

3. CREATE TABLE RESULT AS

SELECT fname,lname,salary

FROM F1

WHERE (F1.photo = M1.i_id);

Query 7: The media data is referenced by the description "big nose" of EMPLOYEE.photo and the description "blast off" of EMPLOYEE.voice. Query decomposition and join operations are needed. The query will break into four subqueries as follows:

1. CREATE TABLE F1 AS

   SELECT all

   FROM EMPLOYEE;

2. CREATE TABLE M1 AS

   SELECT i_id FROM PHOTO1

   WHERE CONTAIN (PHOTO1, 'big nose');

3. CREATE TABLE M2 AS

   SELECT i_id FROM VOICE1

   WHERE CONTAIN (VOICE1, 'blast off');

4. CREATE TABLE RESULT AS

   SELECT fname,lname, photo

   FROM F1

   WHERE (F1.photo = M1.i_id) and (F1.voice = M2.s_id);

Query 8: Media data is referenced by the description "big nose" of EMPLOYEE.photo and the description "pyramid style" of DEPT.dphoto. Query decomposition and a join operation are needed. The query will break into four subqueries as follows:

1. CREATE TABLE F1 AS

   SELECT all FROM EMPLOYEE, DEPT

   WHERE EMPLOYEE.dnum = DEPT.dno;

2. CREATE TABLE M1 AS

   SELECT i_id

   FROM PHOTO1

WHERE CONTAIN (PHOTO1, 'big nose');

3. CREATE TABLE M2 AS

SELECT i_id

FROM DPHOTO2

WHERE CONTAIN (DPHOTO2, 'pyramid style');

4. CREATE TABLE RESULT AS

SELECT fname,lname, dname

FROM F1

WHERE (F1.photo = M1.i_id) and (F1.dphoto = M2.i_id);

After the system gets the final result which is an INGRES relation, the system will generate a cursor called cursored_output to print out the data one tuple at a time. If the output contains any media data, the resulting table does not show us this fact. The system must check the MDBMS catalog information to verify this in order to handle the media data properly. The process of creating and using the cursor is as follows:

EXEC SQL CREATE CURSOR cursor_output AS

SELECT all

FROM RESULT ;

EXEC SQL FETCH CURSOR cursor_output;

print formatted data;

EXEC SQL CREATE CURSOR cursor_output AS

SELECT media data

FROM RESULT ;

EXEC SQL FETCH CURSOR cursor_output;

display the media data;

Where RESULT is the resulting relation or table of a given query.

Our current design and implementation only support SQL selections of the kind given in the above queries. The complex selections such as a nesting condition, or multiple selections are not allowed. Multiple selection conditions can occur with ***and*** and ***or*** Boolean operators. These conditions have to be in the disjunctive normal form. The details of these operations will be discussed in Chapter 4 of this thesis.

IV. IMPLEMENTATION OF THE SYSTEM

A. INTERFACE DESIGN

In section III-D we discussed the retrieval operation using an extended SQL language. While conceptually formatted data and media data are processed as discussed, a decision was made not to use extended SQL as a user interface. This decision was made because a demonstration of the concept of multimedia data processing does not depend on this language and the additional effort required to implement the extended SQL does not provide any additional benefit. Consequently the decision to use an interactive interface with a great deal of user prompts was made. Such an interface is believed to be actually easier for casual users to use. Rather than describing the interface in an abstract manner, I shall describe it with examples.

Consider the following query in extended SQL form as given by a user:

```
SELECT EMPLOYEE.lname, EMPLOYEE.fname, EMPLOYEE.photo, DEPT.dname
    FROM EMPLOYEE, DEPT
    WHERE ((EMPLOYEE.salary  > 20000 and EMPLOYEE.lname = 'wilson' and
                CONTAIN (EMPLOYEE.photo, "big nose")) or
                (CONTAIN (DEPT.dphoto, "pyramid style") ) and
                (EMPLOYEE.dnum=DEPT.dno);
```

When the user wants to specify such a query in the MDBMS, the person shall first choose the retrieval option from the main menu (as in Figure 13 option 3). The system then responds with appropriate instructions step-by-step. Each time when the user's response is entered, the system will return to ask for the next piece of information. The following operations are thus required to complete the above query (the Italics represent the user's responses.):

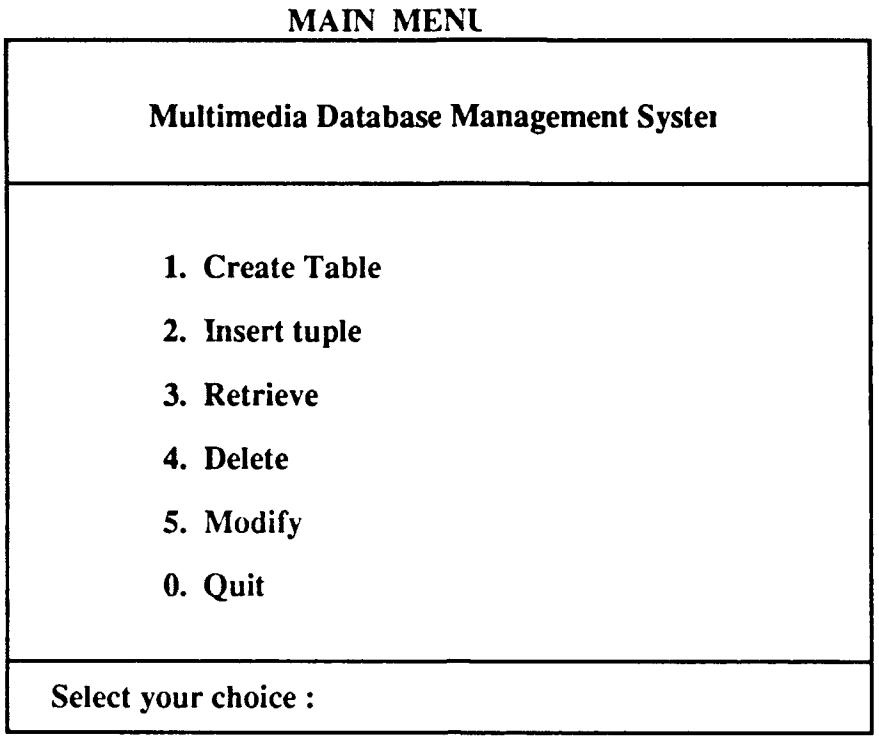

Figure 13 Main menu of the MDBMS

Enter the relation(s) that you want separated by (,)

EMPLOYEE, DEPT <cr>

Please specify the join condition

EMPLOYEE.dnum = DEPT.dno <cr>

Enter the attribute(s) that you want from relation "EMPLOYEE" separate by <,>

fname, lname, photo, voice <cr>

Enter the attribute(s) that you want from relation "DEPT" separate by <,>

dname <cr>

Condition (y/n) *y* <cr>

Group Condition (y/n) *y* <cr>

Enter the relation name *EMPLOYEE*

Enter the attribute name *salary*

Enter the condition

> 20000 <cr>

End group (y/n) *n*

Enter the relation name *EMPLOYEE*

Enter the attribute

photo <cr>

Enter the image description

"big nose"

End group ? *y*

More condition ? *y*

Enter the relation name *DEPT*

Enter the attribute *dphoto <cr>*

Enter the image description

"pyramid style"

End group ? *y*

More condition ? *n*

The multiple selection conditions must be represented in disjunctive normal form, by using the Boolean operator ***and*** inside each group, and Boolean operator ***or*** between groups (In the above illustration the term group condition is used). The idea of using the disjunctive normal form is to simplify implementation without sacrificing functionality and usability. The term group condition means the multiple conditions grouped together and each group may have a single or multiple conditions. For example, the conditions in

```
WHERE ((EMPLOYEE.salary > 20000 and EMPLOYEE.lname = 'wilson'
        and CONTAIN (EMPLOYEE.photo, "big nose")) or
        (CONTAIN (DEPT.dphoto, "pyramid style") ) and
        (EMPLOYEE.dnum=DEPT.dno);
```

are composed of two groups. The first group is ((EMPLOYEE.salary > 20000 and EMPLOYEE.lname = 'wilson' and CONTAIN (EMPLOYEE.photo, "big nose")). The second group is (CONTAIN (DEPT.dphoto, "pyramid style"). The join condition is (EMPLOYEE.dnum = DEPT.dno).

Thus instead of having the users input the whole query all at once in the SQL form, the system will ask to the user for input interactively. When the user enters the conditions into the query, the conditions will be kept in the same group until the user selects the end group condition. The MDBMS then queries the user if he has more conditions or not. If not, then the DBMS will process the conditions that have already been entered. These condition will be decomposed into disjunctive normal form as illustrated.

When the above query is executed, the system will display the formatted data portion as shown below and query the user if a display of image or playing the audio data is desired. The system will respond according to the user's response. The following is therefore the continuation of the display at the computer terminal:

Display of the formatted data

TID	fname	lname	dname
1	Ralph	West	CS
2	Paul	Wilson	CS

Which tuple's image do you want to see ? *1*

Do you want to display that image ? (y/n) *y*

Description : big nose, smiley face.

Quit image window to continue

Do you want to retrieve more images ? (y/n) *n*

(After finishing the first media the user may go to a second one)

Which tuple's sound do you want to hear ? *1*

Do you want to hear the sound ? (y/n) *y*

(The sound is play back for the user)

Do you want to retrieve additional data ? (y/n) *n*

If the user query selects more than one media data, the system will prompt the user for the next media in the selection list, as shown above. When the answer is y, the system will again query which image or sound data is to be displayed or played. When the answer

is n, the system will return to the main menu as no other media data selection is wanted by the user. The user can proceed further to perform other desired operations.

B. QUERY PROCESSING

In the previous chapter, the various cases in which an extended SQL query must be decomposed into multiple SQL queries are illustrated. This method of decomposition requires the generation of temporary relational tables for further processing. Thus in the above example the system will generate the temporary tables, M1, M2, G1, G1 and RESULT through the following SQL statements passed to INGRES:

```
M1 = SELECT i_id
        FROM PHOTO1
        WHERE CONTAIN (PHOTO1, 'big nose');
M2 = SELECT i_id
        FROM DPHOTO2
        WHERE CONTAIN (DPHOTO2, 'pyramid style');
G1 = SELECT EMPLOYEE.lname,EMPLOYEE.fname,EMPLOYEE.photo, DEPT.dname
        FROM EMPLOYEE, DEPT
        WHERE ((EMPLOYEE.salary > 20000 and
            EMPLOYEE.lname = 'wilson' and
            (EMPLOYEE.photo in (SELECT i_id FROM M1)) and
            (EMPLOYEE.dnum=DEPT.dno);
G2 = SELECT EMPLOYEE.lname,EMPLOYEE.fname,EMPLOYEE.photo, DEPT.dname
        FROM EMPLOYEE, DEPT
        WHERE ((EMPLOYEE.salary > 20000 and
                EMPLOYEE.lname = 'wilson' and
                (DEPT.dphoto in (SELECT i_id FROM M2)) and
                (EMPLOYEE.dnum=DEPT.dno);
```

RESULT = SELECT ALL FROM G1 UNION G2;

Note that, in order to search the media data contents, descriptions from many media data tuples must be processed. Prolog is used for this purpose. Natural language descriptions are handled by means of a parser which transforms the descriptions into Prolog predicates and literals to be deposited in a file named "imagei_image_facts" to be used by Prolog. When the temporary tables are obtained, join operation on the media attributes (e.g. photo in the EMPLOYEE relation, now in G1, and the i_id column of the image media relation) is performed to produce the desired final result. The join operation will generate the temporary RESULT relation and a cursor is declared to print out the result out of the RESULT relation. The system will look for attributes which are media data type and will print out a message to ask the user whether to display or play that media, as illustrated in the above example.

C. DATA STRUCTURE FOR RETRIEVAL OPERATION

Having the catalog tables as described in the previous chapter is not sufficient to perform query processing. As shown in the previous sections in this chapter, the retrieval operation actually requires a compiler action to compile the user input into SQL statements for INGRES. Additional tables are therefore required to keep the various information for the purpose of the retrieval operation. The example of the query stated at the beginning of this chapter can be used to illustrate this.

First we need a table to hold the information for selection. In order to process the query, the system needs information on the table name, the attribute names and their data types for each SELECT operation. The table, Selection_Array, is created for this purpose and it has the structure to hold this information as shown in Figure 14.

	1	2	3	4	5
T_name	EMPLOYEE	EMPLOYEE	EMPLOYEE	DEPT	
A_name	fname	lname	photo	dname	
Att_type	formatted	formatted	image	formatted	

Figure 14 Selection_Array table

The second structure is the Condition_Array table. This structure holds the conditions for the query and it contains the table name (Table_name), attribute name (Attribute_name), and the value condition (condition) for each selection condition, as shown in Figure 15. The conditions are entered into the table in the order received from the user.

	1	2	3	4
Table_name	EMPLOYEE	EMPLOYEE	EMPLOYEE	DEPT
Attribute_name	salary	lname	photo	dphoto
Condition	> 2000	='Wilson'	'big nose'	= 'pyramid style'

FIGURE 15 Condition_Array table

The third structure is the Group_Array table. This structure holds the index to the Condition_Array table for each group in the query and it contains the beginning of the group condition (begin_group), and the ending of the group condition (end_group) as contained in the Condition_Array table. Figure 16 is the result that corresponds to the

example query used for illustration, stating that the first three conditions (1-3) belong to one group and the fourth condition belongs to the other group.

begin_group	1	4		
end_group	3	4		

FIGURE 16 Group_Array table

To support media data not accept by INGRES, the system transforms user queries into embeded C SQL statements to be processed by INGRES. This works well when all the conditions and variables are completely defined prior to the compilation of an application program done by INGRES. Unfortunately in our circumstances, the MDBMS receives information from the users only at run time. Since INGRES SQL does not support host variables and INGRES considers the programs in MDBMS as application programs, information from the users at run time cannot be passed to INGRES via the embeded C SQL statements. To solve this problem, we have to modify the C code generated by IN ES in the precompilation process, when SQL statements have already been transformed into C code, in such a way that variables can assigned values at run time. The result is then compiled by the C-compiler for execution.

D. PROGRAM STRUCTURE FOR RETRIEVAL OPERATION

The database operations are written in the programming language C and are separated into 5 submodules as followed:

1. The create table module
2. The insertion module
3. The query module
4. The deletion module

5. The update module

Operations 4 and 5 are not implemented at this time and will not be discussed. The other operations are as follows:

1. The create table module is also implemented by Pei and the detailed design and implementation is in [PE90].

2. The insertion module is also implemented by Pei and the detailed design and implementation is in [PE90].

3. The query module is the module that accept the queries from users and access the database and the media data to get the result. It is separated into 4 submodules as follows:

3.1 The main program for retrieving is **procedure retrieve()**. In order to do retrieval, the users will be asked to enter the selection of the table(s) and attribute(s) that they want to retrieve. If a user does not know the names of the tables or attributes, he can use the help prompt by typing '?' to list all the tables and attributes in the catalog before proceeding any further.

3.2 **Procedure process_condition()** is the procedure to process conditions and will accept the table names, attribute names and then check the attribute type from the catalog to see if the attribute values are valid. If the attribute is media data type, then the media condition for this attribute is set to "true". This procedure also checks for the group condition. If it is a group conditions, then the procedure gcondition is invoked. Otherwise the procedure process_query is used.

3.3 **Procedure gcondition()**. For each group, a value for the beginning and ending of each group is assigned. The procedure accepts the conditions given by the user and insert them into the Condition_Array table as specified in Figure 15.

3.4 **Procedure process_icon()**. This process_icon procedure is used to enter the conditions of any media data in the Condition_Array table.

3.5 **Procedure getatttype()**. This procedure will search the catalog to find the attribute type to be returned to the calling program.

3.6 **Procedure p_att()**. This procedure is used when a user needs help to find the attribute names. He just types '?' for printing all the attribute names of the table he selected.

3.7 **Procedure p_table()**. This procedure is used when a user needs help to find the table names. He just types '?' to print all the table names in the database catalog.

3.8 **Procedure process_query()**. After a user completes his input, the MDBMS will display the pseudo extended SQL code on the screen.

3.9 **Procedure process_query2()**. This procedure will determine whether decomposition of a query is needed. If so, the decomposition will be done and subqueries are set up. Intermediate tables are created as necessary and recomposition is done to produce the final result.

3.10 **Procedure display_photo (imageno)**. If a user selects to display the image, this procedure will be invoked. The image corresponding to the image number (imageno) from the calling program will be displayed.

3.11 **Procedure play_sound (soundno)**. If a user selects to play the sound, this procedure will be invoked. The sound corresponding to the sound number (soundno) from the calling program will be played.

The detail of this code is in APPENDIX B of this thesis.

E. HOW TO LINK AND RUN THE SYSTEM

The system is built on the SUN workstation under the server name Virgo at NPS.CS.NAVY.MIL under account /n/virgo/mdbms/mdbms/demos2. Demos2 is an object code module ready for execution. The program itself is called demos2.sc. It was done with a C precompiler which is listed as ESQLC demos2.sc. This ESQLC will produce demos2.c. After we get demos2.c, we have to compile this program into an object program and link it to the INGRES library and the other subprograms which include ISfunctions.o, ISsubroutines.o, comcprolog.o, suntools library, sunwindows library, and sunpixrects library. The other files that are needed in the same directory are parser6, imagei_image_dicts and imagei_image_facts. To make this link process simpler, a macro Makefile, is used and is given as follows:

```
#
OBJMODS = ISfunctions.o comcprolog1.o ISsubroutine.o
#ING_HOME = /ingres
demos2: demos2.o $(OBJMODS)
        cc demos2.o -o demos2 \
        /ingres/lib/libqlib /ingres/lib/compatlib \
        $(OBJMODS) \
        -lsuntool -lsunwindow -lpixrect -lm

demos2.c: demos2.sc
        esqlc demos2.sc
```

When a user wants to compile and link a new implementation of the demos2, he just types "make demos2" at the prompt of the Unix operating system. The execution module will be named demos2.

V. CONCLUSIONS AND SUMMARY

Many applications requires the use of both formatted and media data. The handling of multimedia data imposes new requirements on database management systems, especially when the integrated support of conventional and multimedia data is needed. In this thesis, an approach to integrating conventional alphanumeric and multimedia data is achieved using the abstract data type concept. We use the INGRES relational DBMS for maintaining the conventional standard data and the media relations.

This thesis outlined some sample applications in which multimedia data is required and presented a design of the system to support the various database operations. Specifically it showed how the catalog information is stored for the processing of both formatted and multimedia data and how the retrieval operation for these data can be achieved by decomposing a user query into multiple subqueries, the results of which are recomposed to form the final result. To achieve our goal, additional tables must be designed to store the various kinds of information for a single selection operation. Many examples were presented throughout the thesis to illustrate our points.

Although conceptually a SQL-like query interface is assumed, an interactive interface was implemented for the system. This was believed to be more usable and simpler to implement. Prompting was used generously. A user can work on the system with very little background or knowledge about the system's handling of formatted and media data.

For lack of time not all the database operations have been implemented. Two companion theses [PE90, AT90], done concurrently, provided the support of table creation, data insertion, and sound data management. Image data management was done in some previous works [Th88]. This thesis concentrated on the implementation of the retrieval process. Except for the nested queries, nearly all the operations related to the

select statement in SQL can be done with the work done in this thesis. To process a single select statement in SQL involving media data, the query must be decomposed into subqueries. Temporary intermediate tables are produced as a result. The final result can only be obtained through the use of join operations. In essence, the retrieval operation as described in this thesis acts like a mini-compiler for an extended SQL query.

The handling of multimedia data with the alphanumeric data in a DBMS is more than just adding new relations into the database. The approach proposed in the MDBMS prototype can retrieve media data based on their contents, described in natural language form. The processing of the descriptions of the media data cannot be done with SQL or in any database systems like INGRES. We had to use a parser and Prolog to process these descriptions.

At present only sound data and image data are supported. However, it is straight forward to extend the capability to handle other media data in a similar manner. The concept of handling both formatted data and media data is amply illustrated through the capability of supporting these two kinds of media data.

Future works will continue on more operations, including the update and delete. The development of a better user interface for the system, the help utility, and transaction processing are planned for the MDBMS prototype.

APPENDIX A

PROGRAM CODE FOR TRANSFORMATION OF IMAGE

Color image can be entered into the MDBMS by first capturing the image with the video camera, then inputing this video signal into an IBM-compatible PC equipped with the Super VIA card[1] to form an image digitize into an image file. This is acheieved by connecting the video output connector in the video camera to the video input (RCA jack) of the Super VIA card, by running the program svu.exe in the PC to capture the image from the video camera, and finally by transferring this file to the SUN system through FTP (File Transfer Protocol). However, the file in the PC must be in the GIF format and the file transfer mode must be set to binary file mode before using FTP. The program giftoras must be invoked to convert the GIF file to SUN raster file format prior to insertion into MDBMS. This can be done by typing the following command at the Sun workstation:

giftoras *(gif filename)* > *(raster filename)*

After the change to raster file we can check by typing showpix followed by raster filename and the picture will be shown on screen. The detail operation of capturing the images with the video camera is given in [PE90]. The program for converting GIF format to raster file format was received from Dr. Klaus Meyer-wegener of University of Erlangen-Nuernberg, Germany, and has been modified to fit into our needed and environment. The following printout is the modified program.

```
/*****************************************
 *          GIF to SUN rasterfile        *
 *****************************************/
/*
 * Usage:
 *      gif2ras <gif-file> > <sun-rasterfile>
 * Compile:
 *       cc gif2ras.c -o gif2ras -lpixrect
```

---

1 Super Video Input adapters products from Jovian Logic Corporation 1990.

```
 */


#include <stdio.h>
#include <rasterfile.h>

char *malloc();
int strncmp();

#define FALSE 0
#define TRUE 1
#define COLSIZE 256
#define PICSIZE 2;
#define SHOWCOUNT 10;


/*
 * Rasterfile variables
 */
struct rasterfile          header;

/*
 * LZW structures and variables
 */
typedef int bool;
unsigned char *stackp;
unsigned int prefix[4096];
unsigned char suffix[4096];
unsigned char stack[4096];
int datasize,codesize,codemask;    /* Decoder working variables */
int clear,eoi;                  /* Special code values */
int avail;
int oldcode;

/*
 * GIF variables
 */
FILE *infile;
unsigned int screenwidth;         /* The dimensions of the screen */
unsigned int screenheight;        /*  (not those of the image)  */
unsigned int rscreenwidth;        /* The dimensions of the raster */
bool global;                   /* Is there a global color map? */
int globalbits;                /* Number of bits of global colors */
unsigned char globalmap[COLSIZE][3];/* RGB values for global color map */
char bgcolor;                  /* background color */
unsigned char *raster;            /* Decoded image data */
unsigned left,top,width,height,rwidth;

char *progname;
char *filename;

void convert();
int  checksignature();
```

```
void readscreen();
int  readimage();
void readextension();
int  readraster();
int  process();
void outcode();
void initraster();
void initcolors();
int  rasterize();
void usage();



main(argc,argv)         /* main program for converting */
int argc;
char *argv[];
{
   extern int optind;
   extern char *optarg;
   int flag;

   progname = *argv;

   while ((flag = getopt(argc, argv, "")) != EOF) {
      switch (flag) {
         default  : fprintf(stderr, "ignoring unknown flag %c\n", flag);
                  usage();
      }
   }

   if (optind >= argc) {
      filename = "stdin";
      convert();
   }
   else {
      filename = argv[1];
      if ((infile = fopen(filename,"r")) == NULL) {  /* test for open input file name */
         perror(filename);
         exit(1);
      }
      convert();
      fclose(infile);
   }

#if defined(ARGS)
  else
      while (optind < argc) {
         filename = argv[optind];
         optind++;
         if ((infile = fopen(filename,"r")) == NULL) {   /* test for open input file name */
            perror(filename);
            continue;
```

```
        }
        convert();
        fclose(infile);
    }
#endif

} /* main */


/* start converting GIF to SUN raster file */
void convert()
{
  char ch;

/*   fprintf(stderr, "%s:\n", filename); */
  if (checksignature())
    return;
  readscreen();
  while ((ch = getc(infile)) != ';' && ch != EOF) {
    switch (ch) {
      case '\0': break; /* this kludge for non-standard files */
      case ',': if (readimage())
                  return;
                break;
      case '!': readextension();
                break;
      default:  fprintf(stderr, "illegal GIF block type\n");
                return;
                break;
    }
  }

} /* convert */

/* Check for the GIF file (GIF file has the signature GIF87 at
  the beginining of the file header */
checksignature()
{
  char buf[6];

  fread(buf,1,6,infile);
  if (strncmp(buf,"GIF",3)) {
    fprintf(stderr, "file is not a GIF file\n");
    return 1;
  }
  if (strncmp(&buf[3],"87a",3)) {
    fprintf(stderr, "unknown GIF version number\n"); /* GIF87 is the GIF version */
    return 1;
  }
  return 0;

} /* checksignature */
```

```
/*
 * Get information which is global to all the images stored in the file
 */

void readscreen()
{
   unsigned char buf[7];

   fread(buf,1,7,infile);
   screenwidth = buf[0] + (buf[1] << 8);
   rscreenwidth = screenwidth + screenwidth%2; /* compensate odd widths */
   screenheight = buf[2] + (buf[3] << 8);
   global = buf[4] & 0x80;
   if (global) {
      globalbits = (buf[4] & 0x07) + 1;
      fread(globalmap,3,1<<globalbits,infile);
   }
   bgcolor = buf[5];
/*
   fprintf(stderr, "    global screen: %dx%dx%d, backgroundcolor: %d\n",
           screenwidth, screenheight, 1<<globalbits, bgcolor);
*/

} /* readscreen */


/* Read the image file and check the colormap */

readimage()
{
   unsigned char buf[9];
   bool local, interleaved;
   char localmap[256][3];
   int localbits;
   register row;
   register i;

   if (fread(buf, 1, 9, infile) == 0) {
      perror(filename);
      exit(1);
   }
   left = buf[0] + (buf[1] << 8);
   top = buf[2] + (buf[3] << 8);
   width = buf[4] + (buf[5] << 8);
   rwidth = width + width%2; /* compensate odd widths */
   height = buf[6] + (buf[7] << 8);
   local = buf[8] & 0x80;
   interleaved = buf[8] & 0x40;
/*
   fprintf(stderr, "    image: %dx%d %s  org: %d,%d\n", width, height,
         interleaved ? "interleaved" : "", left, top);
```

```
*/
  if (local == 0 && global == 0) {
     fprintf(stderr, "no colormap present for image\n");
     return 1;
  }
  if ((raster = (unsigned char*) malloc(rwidth*height)) == NULL) {
     fprintf(stderr, "not enough memory for image\n");
     return 1;
  }
  if (readraster(width, height))
     return 1;

  if (local) {
     localbits = (buf[8] & 0x7) + 1;
/*
     fprintf(stderr, "   local colors: %d\n", 1<<localbits);
*/
     fread(localmap, 3, 1<<localbits, infile);
     initraster(1<<localbits);
     initcolors(localmap, 1<<localbits, bgcolor);
  } else if (global) {
     initraster(1<<globalbits);
     initcolors(globalmap, 1<<globalbits, bgcolor);
  }

  rasterize(interleaved, raster);
  free(raster);

  return 0;

} /* readimage */



/*
 * Read a GIF extension block (and do nothing with it).
 */

void readextension()
{
  unsigned char code;
  int count;
  char buf[255];

  code = getc(infile);
  while (count = getc(infile))
     fread(buf, 1, count, infile);

} /* readextension */



/*
```

```
 * Decode a raster image
 */

readraster(width, height)
unsigned width,height;
{
        unsigned char *fill = raster;
        unsigned char buf[255];
        register bits=0;
        register unsigned datum=0;
        register unsigned char *ch;
        register int count, code;

        datasize = getc(infile);
        clear = 1 << datasize;
        eoi = clear + 1;
        avail = clear + 2;
        oldcode = -1;
        codesize = datasize + 1;
        codemask = (1 << codesize) - 1;
        for (code = 0; code < clear; code++) {
           prefix[code] = 0;
           suffix[code] = code;
         }
        stackp = stack;
        for (count = getc(infile); count > 0; count = getc(infile)) {
           fread(buf,1,count,infile);
           for (ch=buf; count-- > 0; ch++) {
              datum += *ch << bits;
              bits += 8;
              while (bits >= codesize) {
                 code = datum & codemask;
                 datum >>= codesize;
                 bits -= codesize;
                 if (code == eoi) {              /* end of image */
                    goto exitloop;              /* because some GIF files*/
                 }                              /* aren't standard */
                 if (process(code, &fill)) {
                    goto exitloop;
                 }
              }
           }
           if (fill >= raster + width*height) {
              fprintf(stderr, "raster full before eoi code\n");
              goto exitloop;
           }
        }
exitloop:
        if (fill != raster + width*height)  {
           fprintf(stderr, "warning: wrong rastersize: %ld bytes\n",
                 (long) (fill-raster));
           fprintf(stderr, "        instead of %ld bytes\n",
                 (long) width*height);
```

```
        return 0; /* can still draw a picture ... */
    }

    return 0;

} /* readraster */



/*
 * Process a compression code.  "clear" resets the code table.  Otherwise
 * make a new code table entry, and output the bytes associated with the
 * code.
 */

process(code, fill)
register code;
unsigned char **fill;
{
    int incode;
    static unsigned char firstchar;

    if (code == clear) {
        codesize = datasize + 1;
        codemask = (1 << codesize) - 1;
        avail = clear + 2;
        oldcode = -1;
        return 0;
    }

    if (oldcode == -1) {
        *(*fill)++ = suffix[code];
        firstchar = oldcode = code;
        return 0;
    }

   /* if (code > avail) {
        fprintf(stderr, "code % d to large for %d\n", code, avail);
        return 1;

    }
        */

    incode = code;
    if (code == avail) {     /* the first code is always < avail */
        *stackp++ = firstchar;
        code = oldcode;
    }
    while (code > clear) {
        *stackp++ = suffix[code];
        code = prefix[code];
    }
```

```
        *stackp++ = firstchar = suffix[code];
        prefix[avail] = oldcode;
        suffix[avail] = firstchar;
        avail++;

        if (((avail & codemask) == 0) && (avail < 4096)) {
           codesize++;
           codemask += avail;
        }

        oldcode = incode;
        do {
           *(*fill)++ = *--stackp;
        } while (stackp > stack);

        return 0;

} /* process */

/* init raster row and column size */

void initraster(numcols)
int numcols;
{

  header.ras_magic= 0x59a66a95;
  header.ras_width= rwidth;
  header.ras_height= height;
  header.ras_depth= 8;
  header.ras_length= rwidth * height;
  header.ras_type= RT_STANDARD;
  header.ras_maptype= RMT_EQUAL_RGB;
  header.ras_maplength= 3*numcols;

  if (fwrite(&header, sizeof(header), 1, stdout) == 0) {
        perror(progname);
        exit(1);
  }


} /* initraster */



/*
 * Convert a color map (local or global) to arrays with R, G and B
 * values. Pass colors to SUNVIEW and set the background color.
 */

void initcolors(colormap, ncolors, bgcolor)
unsigned char colormap[COLSIZE][3];
int ncolors;
```

```
int bgcolor;
{
   register i;
   unsigned char red[COLSIZE];
   unsigned char green[COLSIZE];
   unsigned char blue[COLSIZE];

/*
fprintf(stderr, "          ");
*/
   for (i = 0; i < ncolors; i++) {
      red[i]   = colormap[i][0];
      green[i] = colormap[i][1];
      blue[i]  = colormap[i][2];
/*
fprintf(stderr, "  %3u: %3u, %3u, %3u", i, red[i], green[i], blue[i]);
if (i%3 == 2) fprintf(stderr, "\n          ");
*/
   }

/*
fprintf(stderr, "\n     Background: %3u: %3u, %3u, %3u\n", bgcolor,
                  red[bgcolor], green[bgcolor], blue[bgcolor]);
*/

 if (fwrite(red, 1, ncolors, stdout) == 0) {
        perror(progname);
        exit(1);
 }
 if (fwrite(green, 1, ncolors, stdout) == 0) {
        perror(progname);
        exit(1);
 }
 if (fwrite(blue, 1, ncolors, stdout) == 0) {
        perror(progname);
        exit(1);
 }

} /* initcolors */



/*
 * Read a row out of the raster image and write it to the screen
 */

rasterize(interleaved, raster)
int interleaved;
register unsigned char *raster;
{
   register row, col;
   register unsigned char *rr;
   unsigned char * newras;
```

```
#define DRAWSEGMENT(offset, step)                    \
      for (row = offset; row < height; row += step) { \
         rr = newras + row*rwidth;                    \
         bcopy(raster, rr, width);                    \
         raster += width;                             \
      }


   if ((newras = (unsigned char*) malloc(rwidth*height)) == NULL) {
      fprintf(stderr, "not enough memory for image\n");
      return 1;
   }
   rr = newras;
   if (interleaved) {
      DRAWSEGMENT(0, 8);
      DRAWSEGMENT(4, 8);
      DRAWSEGMENT(2, 4);
      DRAWSEGMENT(1, 2);
   }
   else
      DRAWSEGMENT(0, 1);

   if (!fwrite(newras, 1, rwidth*height, stdout)) {
      perror(progname);
      exit(1);
   }

   free(newras);


} /* rasterize */


/* print error for usage error */
void usage()
{

      fprintf(stderr, "usage: %s [-l<num] [-s<num>] [-b] gif-files\n",
                  progname);

} /* usage */
```

APPENDIX B

PROGRAM CODE FOR MULTIMEDIA DATA RETRIEVAL MANAGEMENT

```
/*****************************************************************************/
/*****************************************************************************/
/* MDBMS                                                                     */
/* The query interface of the multimedia database management system          */
/* Date: 19 Sep 1990                                                         */
/* Modify Date:                                                              */
/*****************************************************************************/
/*****************************************************************************/
/* The purpose for this program is to demonstrate the prototype of the       */
/* Multimedia Database Management System                                     */
/*****************************************************************************/
/*****************************************************************************/
#include <stdio.h>
#include <string.h>
#include <pixrect/pixrect_hs.h>
#include <sys/wait.h>
#include <suntool/sunview.h>
#include <suntool/canvas.h>
/* For sound module had to include the socket file                           */
# include <sys/types.h>      /* Sound module */
# include <sys/socket.h>     /* Sound module */
# include <netinet/in.h>     /* Sound module */
# include <netdb.h>          /* Sound module */
# include "snd_errs.c"
/* To connect to the INGRES DBMS we have to set commmunication area          */
# include "/ingres/files/eqsqlca.h"
  static IISQLCA sqlca = {0}; /* SQL Communications Area */
#define NOT_FOUND 100         /* Not found for the search */
#define FILENAMELEN 64              /* Max for filename is 64 */
#define DESCRLEN 500        /* Define the description data to 500 char */
#define ERRMLEN 70
#define DESCR_WORD_ERR -30000              /* The parser check for error code */
#define DESCR_STRUCTURE_ERR -30001         /* The parser check for error code */
#define QUERY_WORD_ERR -30002              /* The parser check for error code */
#define QUERY_STRUCTURE_ERR -30003         /* The parser check for error code */
#define DESCR_TOO_LONG_ERR -30004          /* The parser check for error code */
#define PROGRAM_ERR 400                    /* The parser check for error code */
#define NAME_LENGTH 13
#define ERROR_FREE 0
#define SOUND_ERROR -1
#define TRUE 1              /* Defined for create & insert operation */
#define FALSE 0             /* Defined for create & insert operation */
#define MAX_TABLE 20          /* Defined for create & insert operation */
#define MAX_ATT 200          /* Defined for create & insert operation */
#define MAX_PATH 64           /* Defined for create & insert operation */
#define MAX_PHRASE 127          /* Defined for create & insert operation */
```

```
#define MAX_DESCRP 500        /* Defined for create & insert operation */
#define NOT_FOUND 100         /* Defined for create & insert operation */
/* Structure for the sound header file used to get the registration datum */
/* when insert a sound media into database                                */
typedef struct SND_HDR {
                    char sfname[13];
                    int s_size;
               int s_samplrate;
                    int s_encoding;
                    float s_duration;
               int s_resolution;
               };
struct SND_HDR s_hdr;
char c;                          /* For catrige return only */
typedef char STR_name[13];       /* For both table name and att name */
typedef char STR_value[21];      /* For all vales of data type c20   */
typedef char STR_path[MAX_PATH+1];  /* The f_id of media records      */
typedef char STR_descrp[MAX_DESCRP+1];/* The description of media record */
/* Structure for the table catalog, used to get information from text file*/
/* "dbtable" which hold the standard relations in MDBMS                  */
typedef struct table {
                              STR_name table_name;
                              int table_key;
                              int att_count;
                              int att_entry;
                              };
struct table table_array[MAX_TABLE]; /* Relation table in database */
int table_index;          /* Next available index of table_array */
int table_list[MAX_TABLE]; /* Integer array hold the index of table_array */
int table_count = 0, /* # of index (relation) in table_list */
   table_cursor= 0, /* Current index of table_list */
   table_entry = 0; /* Current index of table_list which get */
                    /* by the function check_table_name()!!  */
/* Structure for the attribute catalog, used to get information from text */
/* file "dbatt" which hold all attributes exist in MDBMS and grouped     */
/* together associate to each relation from 1st att to last att          */
typedef struct att  {
                              STR_name att_name;
                              STR_name data_type;
                              int media_id;      /* Next available ID */
                              int next_index;
                              int value_entry;
                              };
struct att att_array[MAX_ATT]; /* All the att_name in database */
int att_index = 0,  /* Next available index of att_array */
   att_cursor = 0,  /* Current index of att_array */
   att_count = 0;   /* # of attribute entered during creation */
STR_name data_type; /* Global string variable */
char table_name[40]; /* Global string variable for temterary read in */
char att_name[40];  /* Global string variable for temterary read in */
                    /* Declare more to avoid bus error              */
int act_media_list[10]; /* Active index of media att_name in operation */
int act_media_count;    /* # of index in act_media_list */
STR_name media_name;    /* Global string variable used to generate */
                        /* the unique media table name in database */
```

```
int table_key; /* Append key for the media attribute name in that table */
int img_value[20],snd_value[20],i_value[20]; /* Data value arrays */
float f_value[20];                    /* Data value arrays */
STR_value c_value[20];                /* Data value arrays */
int img_index = 0,                    /* Indices of data value arrays */
   snd_index = 0,                     /* Indices of data value arrays */
   i_index = 0,                       /* Indices of data value arrays */
   f_index = 0,                       /* Indices of data value arrays */
   c_index = 0;                       /* Indices of data value arrays */
/* Structure to hold whole tuple values in image media relation    */
typedef struct img {
                              int i_id;
                              STR_path f_id;
                              STR_descrp descrp;
                              int height;
                              int width;
                              int depth;
                              };
struct img img_record[20];   /* Values of image media relation */
/* Structure to hold whole tuple values in sound media relation */
typedef struct snd {
                              int s_id;
                              STR_path f_id;
                              STR_descrp descrp;
                              int size;
                              int samp_rate;
                              int encoding;
                              float duration;
                              int resolution;
                              };
struct snd snd_record[20]; /* Values of sound media relation */
STR_descrp descrp;         /* Global for insert tuple operation */
FILE *img_file, *snd_file; /* Global for insert tuple operation */
typedef struct group {     /* begin and end group for condition */
                              int begingroup;
                              int endgroup;
                              };
char join_condition[100];
typedef struct select_att {  /* selection attribute */
          STR_name t_name;
          STR_name a_name;
          STR_name data_type;
          int media_type;
          };
int look_more=0;             /* use for loop the cursor */
typedef struct select_tab {
          STR_name t_name;
          int tab_index;
          };
struct select_att satt[10];
struct select_tab stab[10];
struct group group_count[10];
int o,p,k,numcon,numgroup,icond;
STR_name tab[10];
char condition[100];
```

```
/* Selection attribute */
/* Condition attribute */
STR_name att[10];
/* Each group of attribute */
int att_group[10];
/* Condition type of each attribute 0 for formatted 1 for image 2 for sound*/
int contype[10];
/* Media attribute for description */
STR_name media_att[10];
int number_media;
/* Condition for each attribute */
char con[10][100];
/* Attribute type for each select */
STR_name atttype[10];
int  cond,gcond,i_cond[10],m=0,x=0,y=0,n=0,o=0;
char buff[100],a,yes_no_answer();
/*****************************************************************************/
/* Get yes or no answer from user                                            */
/*****************************************************************************/
char yes_no_answer()
{
  char answer = '?';
  answer = getchar();
  while (!(answer == 'y' || answer == 'n'))
   {
    printf("\nPlease answer y for yes or n for no::");
    answer = getchar();
    while ((c =getchar()) != '\n')
     ;
   }
  getchar(); /* To let the next gets() works properly and nothing else */
  return (answer);
} /* End of yes_no_answer() */
/*****************************************************************************/
/* To clear screen                                                           */
/*****************************************************************************/
void clr_scr()
{
          putchar('\033');
          putchar('[');
          putchar('H');
          putchar('\033'),
          putchar('[');
          putchar('J');
} /* End of clr_scr() */
/*****************************************************************************/
/* Assign -1 to next_index in the last att_name to indicate the end of list*/
/*****************************************************************************/
void assign_end_mark()
{
 int i = 0,
    last_index = 0;
 for (i = 0; i < table_count; i++)
 {
  last_index += table_array[i].att_count;
```

```
  att_array[last_index-1].next_index = -1; /* assign end mark here */
 } /* End of for loop */
} /* End of assign_end_mark() */
/*****************************************************************************/
/* Send command from SUN to PC to play the SOUND media file          */
/*****************************************************************************/
play_sound(filename)
char *filename;
{
        char   *pcname="pclum2"; /* Remote PC host name */
        short   port = 2000;     /* Virtual port number between SUN & PC */
        int sock;
        struct sockaddr_in server;
        struct hostent *hp, *gethostbyname();
        char buf[1024];
        /* Create socket */
        sock = socket(AF_INET, SOCK_STREAM, 0);
        if (sock < 0) {
                perror("opening stream socket");
                return;
        }
        /* Connect socket using name specified by command line. */
        server.sin_family = AF_INET;
        hp = gethostbyname(pcname);
    if (hp == 0) {
                fprintf(stderr, "%s: unknown host\n", pcname);
                return;
        }
        bcopy((char *)hp->h_addr, (char *)&server.sin_addr, hp->h_length);
        server.sin_port = htons(port);
        if (connect(sock,
                (struct sockaddr *)&server, sizeof server ) < 0) {
                perror("connecting stream socket");
                return;
        }
        if (write(sock,filename,12) < 0)   /*gets the filename for playing*/
                perror("Writing on stream socket");
        close(sock);
        return;
}
/*****************************************************************************/
/* Get the header information from the sound text file which is already   */
/* sent from PC to SUN                                                */
/*****************************************************************************/
snd_load(filename)
char *filename;                 /* Given input text file */
{
 FILE  *f;
 if ((f = fopen(filename,"r")) == NULL)      /* open for reading */
           {
            displayerr(ROPEN);
            return SOUND_ERROR;
           }
 /* ***** read the header from the predesignated input file  */
          fscanf(f,"%s",s_hdr.sfname);
```

```
            fscanf(f,"%d",&s_hdr.s_size);
            fscanf(f,"%d",&s_hdr.s_samplrate);
            fscanf(f,"%d",&s_hdr.s_encoding);
            fscanf(f,"%f",&s_hdr.s_duration);
            fscanf(f,"%d",&s_hdr.s_resolution);
            fclose(f);
  return;
}
/*****************************************************************************/
/* Load catalog datum from 3 files: "dbtable", "dbatt" and "dbkey"       */
/*****************************************************************************/
void load_data()
{
 FILE *f, *g, *h;
 STR_name dummy;
 int entry=0,
     i=0;
 f = fopen("dbtable","r");        /* Read the table for catalog into memory */
 if(!feof(f))
   fscanf(f,"%s",dummy);          /* Skip the first dummy line in file */
 while (!feof(f))
   {
    fscanf(f,"%s%d%d", table_array[table_index].table_name,
                       &table_array[table_index].table_key,
                       &table_array[table_index].att_count);
    table_array[table_index].att_entry = entry;
    entry += table_array[table_index].att_count;
     table_index ++;
   }
 fclose(f);                       /* close the input file */
 table_count = table_index;
 if (table_count != 0) /* i.e. database is NOT empty */
 {
   for (i = 0; i < table_count; i++ )
     table_list[i] = i;
  g = fopen("dbatt","r"); /* Read the attribute file to catalog in memory */
  if(!feof(g))
    fscanf(g,"%s",dummy); /* Skip the first dummy line in file */
  while (!feof(g))
   {
    fscanf(g,"%s%s%d",att_array[att_index].att_name,
                      att_array[att_index].data_type,
                  &att_array[att_index].media_id);
    att_array[att_index].next_index = att_index+1;
     att_index++;
   }
  fclose(g);       /* close the attribute file */
  assign_end_mark();
  h = fopen("dbkey","r");
  if(!feof(h))
    fscanf(h,"%s",dummy);        /* Skip the first dummy line in file */
  while (!feof(h))
    fscanf(h,"%d", &table_key); /* Next available table key append to */
  fclose(h);                     /* the end of media att_name is unique */
 }
```

```
 else
  {
   printf("EMPTY DATABASE!!\n\nHit return to continue\n");
   putchar('\007');
   table_key = 1;
   while((c = getchar()) != '\n')
   ; /* Not return do nothing */
  } /* End of if */
} /* End of load_data() */
/*****************************************************************************/
/* Save catalog datum back to 3 files same as above                          */
/*****************************************************************************/
void store_data()
{
 FILE *f, *g, *h;
 STR_name dummy;
 int i = 0,
    j = 0,
    count = 0,
    entry = 0;
 strcpy(dummy, "***dummy***");
 if (table_count > 0)
 {
  f = fopen("dbtable","w");
  fprintf(f,"%s", dummy);
  for (i = 0; i < table_count; i++)
   fprintf(f,"\n%s\t%d\t%d", table_array[table_list[i]].table_name,
                             table_array[table_list[i]].table_key,
                             table_array[table_list[i]].att_count);
  fclose(f);
  g = fopen("dbatt","w");
  fprintf(g,"%s", dummy);
  for (i = 0; i < table_count; i++)
  {
       count = table_array[table_list[i]].att_count;
       entry = table_array[table_list[i]].att_entry;
       for (j = 0; j < count; j++)
       {
         fprintf(g,"\n%s\t%s\t%d", att_array[entry].att_name,
                                   att_array[entry].data_type,
                                   att_array[entry].media_id);
         entry = att_array[entry].next_index;
       } /* End of for loop j */
  } /* End of for loop i */
  fclose(g);
  h = fopen("dbkey","w");
  fprintf(h,"%s", dummy);
  fprintf(h,"\n%d", table_key);
  fclose(h);
 } /* End of if table_count > 0 */
} /* End of store_data() */
/*****************************************************************************/
/* Print out data information on screen (TEMPERARY FOR CHECKING PURPOSE)  */
/*****************************************************************************/
void print_out_data()
```

```
{
  int i = 0,
      j = 0,
      count = 0,
      entry = 0;
  printf("\n");  /* New line */
  for (i = 0; i < table_count; i++)
   printf("%12s\t%d\t%d\t%d\n", table_array[table_list[i]].table_name,
                                table_array[table_list[i]].table_key,
                                table_array[table_list[i]].att_count,
                                table_array[table_list[i]].att_entry);
  while ((c = getchar()) != '\n')
  ;
  for (i = 0; i < table_count; i++)
  {
    count = table_array[table_list[i]].att_count;
    entry = table_array[table_list[i]].att_entry;
     for (j = 0; j < count; j++)
          {
            printf("%12s\t%12s\t%d\n",att_array[entry].att_name,
                                      att_array[entry].data_type,
                                      att_array[entry].media_id,
                                      att_array[entry].next_index);
            entry = att_array[entry].next_index;
          }  /* End of for loop j */
    while ((c = getchar()) != '\n')
     ;
  }  /* End of for loop i */
}  /* End of print_out_data() */
/*****************************************************************************/
/* Get the user choice                                                       */
/*****************************************************************************/
char user_choice()
{
  char answer = '?';
  while (!( '0'<= answer && answer <= '6'))
  {
    clr_scr();
    printf("\n\t\tMultimedia Database Management System\n");
   printf("\t==========================================================\n");
    printf("\n\t1. Create Table");
    printf("\n\t2. Insert Tuple");
    printf("\n\t3. Retrieve");
    printf("\n\t4. Delete");
    printf("\n\t5. Modify");
    printf("\n\t6. Print out current data information(test purpose)");
    printf("\n\t0. Quit\n");
   printf("\t==========================================================\n");
    printf("\n\tSelect your choice :: ");
    answer = getchar();
    while ((c = getchar()) != '\n')
    ;  /* Not return do nothing */
  }  /* End of while */
  return (answer);
}  /* End of user_choice() */
```

```
/*****************************************************************************/
/************************** Start for CREATION ******************************/
/*****************************************************************************/
/*****************************************************************************/
/* Check the table_name if its last char is any digit, which is not allowed*/
/* because the media table is unique across the whole database by appending*/
/* the particular table_key from '0' to '999' in this program          */
/*****************************************************************************/
int check_last_char(c_last)
char c_last;
{
  int found = FALSE;  /* Initialize to false */
  if ('0' <= c_last && c_last <= '9')
    found = TRUE;
  return (found);
} /* End of check_last_char(c_last) */
/*****************************************************************************/
/* Check the table_name if it is duplicate                          */
/*****************************************************************************/
int check_table_name()
{
  int i = 0;
  int found = 0;  /* Initialize to false */
  while ((!(found)) && (i < table_count))
  {
   if (strcmp(table_array[table_index].table_name,
                  table_array[table_list[i]].table_name) == 0)
       {
         found = TRUE;
         table_entry = i;  /* Don't use "table_cursor = i" because  */
       }                   /* table_cursor can't change in the      */
    else                   /* function "change_table_name()"!!      */
       i++;
  } /* End of while */
  return (found);
} /* End of check_table_name() */
/*****************************************************************************/
/* Check the att_name if it is duplicate within the relation in the first  */
/* 9 characters. Because the last 3 characters are used to append the key  */
/*****************************************************************************/
int check_att_name()
{
  int i = 0,
     entry;
  int found = 0;  /* Initialize to false */
  char new_att_name[9],
       exit_att_name[9];
  strncpy(new_att_name, att_array[att_index].att_name, 9);
  new_att_name[9] = '\0';  /* To end of the string */
  entry = table_array[table_list[table_cursor]].att_entry;
  while ((!(found)) && (i < att_count))     /* att_count is global var */
  {
   strncpy(exit_att_name, att_array[entry].att_name, 9);
   exit_att_name[9] = '\0';  /* To end of the string */
   if (strcmp(new_att_name, exit_att_name) == 0)
```

```
            found = TRUE;
        else
            {
              i++;
              entry = att_array[entry].next_index;
            } /* End of if else */
    } /* End of while */
    return (found);
} /* End of check_att_name() */
/*****************************************************************************/
/* Return the data_type which selected from user. We allow c20 as the only */
/* character data type at this time, it could be able to allocate the      */
/* data value array dynamically by mallac to make it more flexible         */
/*****************************************************************************/
void select_data_type()
{
    char answer = '?';
    while ((c = getchar()) != '\n')
    ; /* Not return do nothing */
    while (!( '1'<= answer && answer <= '5'))
    {
      printf("\nSelect::(1)integer (2)float (3)c20 (4)image (5)sound");
      printf("\n\Select your choice :: ");
      answer = getchar();
      while ((c = getchar()) != '\n')
      ; /* Not return do nothing */
    } /* End of while */
    switch (answer)
    {
      case '1':
                strcpy(data_type, "integer");
                break;
      case '2':
                strcpy(data_type, "float");
                break;
      case '3':
                strcpy(data_type, "c20");
                break;
      case '4':
                strcpy(data_type, "image");
                break;
      case '5':
                strcpy(data_type, "sound");
                break;
    } /* Ened of switch */
} /* End of select_data_type() */
/*****************************************************************************/
/* Get the att_name, data_type from user input                               */
/*****************************************************************************/
void get_att_name()
{
    int found = TRUE;
    char set_down = 'n';
    while (found)
    {
```

```
    printf("\nEnter attribute name:(Maximum 12 characters)\n");
    att_name[0] = '\0';
    scanf("%s", att_name);
    if (strlen(att_name) >= 13)  /* Over maximum name length */
         {
           printf("\nSorry!! Attribute Name OVER 12 characters!");
           putchar('\007');
         }
     else
         {
           strncpy(att_array[att_index].att_name, att_name, 12);
           found = check_att_name();
           if (found)
             {
                  printf("The first 9 characters must unique!\n");
                  printf("The duplicate attribute name entered!\n");
                  printf("Invalid attribute name! ENTER AGAIN!!\n");
                  putchar('\007');
             }
           else
             {
                  printf("\nSelect data type of attribute::");
                  while (set_down != 'y')
                   {
                     select_data_type();
                     printf("\nData Type: %s? (y/n)::", data_type);
                     set_down = yes_no_answer();
                   }
                  strcpy(att_array[att_index].data_type, data_type);
             } /* End of if else */
         } /* End of if else */
  } /*End of while */
} /* End of get_att_name() */
/*****************************************************************************/
/* Create a relation table according to the user input                       */
/*****************************************************************************/
void create_table()
{
  char more_att = 'y';  /* More att_name or not */
  int i = 0,
     entry,
     name_len;
  int table_found = TRUE;  /* Initialize to true */
  while (table_found)
   {
   printf("\nEnter table_name:(Maximum 12 characters)\n");
    table_name[0] = '\0';
   scanf("%s", table_name);
   if ((name_len = strlen(table_name)) >= 13) /*Over maximum name length*/
     {
          printf("\nSorry!! Table Name OVER 12 characters!");
          putchar('\007');
     }
    else
     {
```

```
            if (check_last_char(table_name[name_len - 1]))
             {
              printf("Sorry! Please never end a table name with a digit!\n");
              printf("Invalid table name! ENTER AGAIN!!\n");
              putchar('\007');
             }
            else
             {
              strcpy(table_array[table_index].table_name, table_name);
              table_found = check_table_name();
              if (table_found)
               {
                    printf("The duplicate table name entered!\n");
                    printf("Invalid table name! ENTER AGAIN!!\n");
                    putchar('\007');
               }
             } /* End of if else */
       } /* End of if else */
    } /* End of while (found) */
   table_array[table_index].table_key = table_key;
   table_array[table_index].att_entry = att_index;
   table_list[table_count] = table_index;
   table_key++;
   table_cursor = table_count;
   table_count++;
   att_count = 0; /* Initialize as zero at beginning, global in each time */
   while (more_att == 'y')
   {
    get_att_name();
    att_array[att_index].media_id = 1;
    att_array[att_index].next_index = att_index + 1;
     att_index++;
     att_count++;
    printf("\nMore attribute in the table? (y/n)::");
    more_att = yes_no_answer();
   } /* End of while */
   att_array[att_index -1].next_index = -1; /* Assign the end mark */
   table_array[table_index].att_count = att_count;
   table_index ++;
} /* End of create_table() */
/*****************************************************************************/
/* Get the user choice to modify the current table in create operation    */
/*****************************************************************************/
char modify_choice()
{
  char answer = '?';
  getchar(); /* NOTHING but extract out the previous CR */
  while (!(( '0'<= answer && answer <= '5') ||
                              (answer == 'h') || (answer == 'H')))
  {
    printf("\n\t\tModify Table Menu for Creation\n");
    printf("\t==========================================");
    printf("\n\t1. Change Table Name");
    printf("\n\t2. Change Attribute Name");
    printf("\n\t3. Change Data Type");
```

```
    printf("\n\t4. Insert A Attribute");
    printf("\n\t5. Delete A Attribute");
    printf("\n\t0. Quit");
    printf("\n\th or H:: Show current information\n");
   printf("\t=============================================================\n");
    printf("\n\tSelect your choice :: ");
    answer = getchar();
    while ((c = getchar()) != '\n')
    ; /* Not return do nothing */
  } /* End of while */
  return (answer);
} /* End of modify_choice() */
/*****************************************************************************/
/* Print out the current table which the user want to modify          */
/*****************************************************************************/
void print_table()
{
  int i = 0,
     count = 0,
     entry = 0;
  clr_scr();
  entry = table_array[table_list[table_cursor]].att_entry;
  count = table_array[table_list[table_cursor]].att_count;
  printf("\nTable Name:: %s\n",
                    table_array[table_list[table_cursor]].table_name);
  printf("\nOrder\tAttribute Name\t\tData Type\n");
  for (i = 0; i < count; i++)
  {
   printf(" %d \t%13s\t\t%s\n",(i+1) , att_array[entry].att_name,
                                        att_array[entry].data_type);
   entry = att_array[entry].next_index;
  } /* End of for loop i */
} /* End of print_table() */
/*****************************************************************************/
/* Change the current table name which the user want to create        */
/*****************************************************************************/
void change_table_name()
{
  int table_found = TRUE;
  while (table_found)
   {
    printf("\nCurrent Table Name:: %s\n\n",
                    table_array[table_list[table_cursor]].table_name);
    printf("Change to::");
    table_name[0] = '\0';
    scanf("%s", table_name);
    if (strlen(table_name) >= 13) /* Over maximum name length */
        {
         printf("\nSorry!! Table Name OVER 12 characters!");
         putchar('\007');
        }
    else
        {
         strcpy(table_array[table_index].table_name, table_name);
         table_found = check_table_name();
```

```
            if (table_found)
              {
                    printf("\nThe duplicate table name entered!!!\n");
                    printf("\nInvalid table name! ENTER AGAIN!!!\n");
                    putchar('\007');
              };
           } /* End of if else */
    } /* End of while */
  strcpy(table_array[table_list[table_cursor]].table_name,
                                        table_array[table_index].table_name);
  printf("\nNew Table Name:: %s\n\n",
                        table_array[table_list[table_cursor]].table_name);
  while ((c = getchar()) != '\n')
  ;
} /* End of change_table_name() */
/*****************************************************************************/
/* Change the name of current attribute which the user want to create      */
/*****************************************************************************/
void change_att_name()
{
  int i = 0,
      count = 0,
      entry = 0,
      order = 0;
  int found = TRUE;
  print_table();
  printf("Select the order which you want to change its name::\n");
  printf("Any other key to cancel the operation!! Select::");
  scanf("%d", &order);
  entry = table_array[table_list[table_cursor]].att_entry;
  count = table_array[table_list[table_cursor]].att_count;
  if (1 <= order && order <= count)
    {
          for (i = 1; i < order; i++)
           entry = att_array[entry].next_index;
          att_cursor = entry;  /* Assign the current index of att_array */
          while (found)
          {
           printf("\nCurrent Att_Name:: %s\n\n",
                                        att_array[att_cursor].att_name);
           printf("Change to::");
           att_name[0] = '\0';
           scanf("%s", att_name);
           if (strlen(att_name) >= 13)  /* Over maximum name length */
            {
             printf("\nSorry!! Attribute Name OVER 12 characters!");
              putchar('\007');
            }
           else
            {
             strcpy(att_array[att_index].att_name, att_name);
             found = check_att_name();
             if (found)
              {
                    printf("The duplicate attribute name entered!\n");
```

```
                printf("\nInvalid attribute name! ENTER AGAIN!!!\n");
                putchar('\007');
              }
             else
              {
                strcpy(att_array[att_cursor].att_name,
                                att_array[att_index].att_name);
                printf("\nNew Att_Name:: %s\n\n",
                                att_array[att_cursor].att_name);
              } /* End of if else */
            } /* End of if else */
          } /*End of while */
    }
   else
    {
        printf("\nSorry! You entered the wrong order!! Please redo again.\n");
        putchar('\007');
        while ((c = getchar()) != '\n')
        ;
    } /* End of if else */
} /* End of change_att_name() */
/*****************************************************************************/
/* Change the data type of current attribute which the user want to create */
/*****************************************************************************/
void change_data_type()
{
  int i = 0,
     count = 0,
     entry = 0,
     order = 0;
  char set_down = 'n';
  print_table();
  printf("Select the order which you want to change the data type::\n");
  printf("Any other key to cancel the operation!! Select::");
  scanf("%d", &order);
  entry = table_array[table_list[table_cursor]].att_entry;
  count = table_array[table_list[table_cursor]].att_count;
  if (1 <= order && order <= count)
    {
        for (i = 1; i < order; i++)
         entry = att_array[entry].next_index;
        att_cursor = entry; /* Assign the current index of att_array */
        printf("\nCurrent Att_Name:: %s\n",
                                att_array[att_cursor].att_name);
        printf("Current Data_Type:: %s\n",
                                att_array[att_cursor].data_type);
        printf("Change to:: ");
        while (set_down != 'y')
         {
           select_data_type();
           printf("\nData Type: %s? (y/n)::", data_type);
           set_down = yes_no_answer();
         }
        strcpy(att_array[att_cursor].data_type, data_type);
        printf("\nAtt_Name:: %s\n", att_array[att_cursor].att_name);
```

```
        printf("New Data Type:: %s\n",att_array[att_cursor].data_type);
    }
   else
    {
        printf("\nSorry! You entered the wrong order!! Please redo again.\n");
        putchar('\007');
        while ((c = getchar()) != '\n')
        ;
    } /* End of if else */
} /* End of change_data_type() */
/*****************************************************************************/
/* Insert a new attribute before create operation                            */
/*****************************************************************************/
void insert_att()
{
  int i = 0,
      count = 0,
      pre_entry = 0,
      entry = 0,
      order = 0;
  print_table();
  printf("Select the order where new attribute you want be!!\n");
  printf("(Maximum + 1) will add new attribute at the end!!\n");
  printf("Select the new attribute's order::\n");
  printf("Any other key to cancel the operation!! Select::");
  scanf("%d", &order);
  entry = table_array[table_list[table_cursor]].att_entry;
  count = table_array[table_list[table_cursor]].att_count;
  if (1 <= order && order <= (count + 1))
    {
        for (i = 1; i < order; i++)
          {
           pre_entry = entry;
           entry = att_array[entry].next_index;
          }
        get_att_name();
        att_array[att_index].media_id = 1;
        /* Rearrange the link list of attributes in the relation */
        if (order == 1)
          table_array[table_list[table_cursor]].att_entry = att_index;
        else
          att_array[pre_entry].next_index = att_index;
        att_array[att_index].next_index = entry;
        att_index++;
        att_count++;
        table_array[table_list[table_cursor]].att_count = att_count;
    }
   else
    {
        printf("\nSorry! You entered the wrong order!! Please redo again.\n");
        putchar('\007');
        while ((c = getchar()) != '\n')
        ;
    } /* End of if else */
} /* End of insert_att() */
```

```
/*****************************************************************************/
/* Delete a attribute before create operation                                */
/*****************************************************************************/
void delete_att()
{
  int i = 0,
      count = 0,
      pre_entry = 0,
      entry = 0,
      order = 0;
  print_table();
  printf("Select the order of attribute which you want delete::\n");
  printf("Any other key to cancel the operation!! Select::");
  scanf("%d", &order);
  entry = table_array[table_list[table_cursor]].att_entry;
  count = table_array[table_list[table_cursor]].att_count;
  if (1 <= order && order <= count)
    {
          for (i = 1; i < order; i++)
            {
             pre_entry = entry;
             entry = att_array[entry].next_index;
            }
          att_cursor = entry;
          printf("\nDelete %s? (y/n)::", att_array[att_cursor].att_name);
          if (yes_no_answer() == 'y')
            {                          /* Rearrange the link list of */
              if (order == 1)          /* attributes in the relation */
               table_array[table_list[table_cursor]].att_entry
                        = att_array[entry].next_index;
              else
               att_array[pre_entry].next_index
                        = att_array[entry].next_index;
              att_count--;
             table_array[table_list[table_cursor]].att_count = att_count;
            }
          else
            ; /* End of if else */
    }
  else
    {
          printf("\nSorry! You entered the wrong order!! Please redo again.\n");
          putchar('\007');
          while ((c = getchar()) != '\n')
          ;
    } /* End of if else */
} /* End of delete_att() */
/*****************************************************************************/
/* Modify the current table which the user want to create                    */
/*****************************************************************************/
void modify_table()
{
  char answer = '?';
  while (answer != '0')
  {
```

```
    answer = modify_choice();
    switch(answer)
    {
          case '1' :
                    change_table_name();
                    break;
          case '2' :
                    change_att_name();
                    break;
          case '3' :
                    change_data_type();
                    break;
          case '4' :
                    insert_att();
                    break;
          case '5' :
                    delete_att();
                    break;
          case '0' :
                    break;
          case 'H' :
          case 'h' :
                    print_table();
                    break;
    } /* End of switch */
  } /* End of while */
} /* End of modify_table() */
/*****************************************************************************/
/* Display the table information that the user entered before create      */
/*****************************************************************************/
void display_info()
{
  char modify = 'y';
  while (modify == 'y')
  {
    clr_scr();
    print_table();
    printf("\nAny change before create? (y/n)::");
    modify = yes_no_answer();
    if (modify == 'y')
          modify_table();
  } /* End of while */
} /* End of display_info() */
/*****************************************************************************/
/* Get media table name by appending table_key at the end of att_name     */
/*****************************************************************************/
void get_media_name()
{
  int index;   /* Index of string used to append table_key into att_name */
  int i_key,   /* Integer value of table_key */
     key_no,   /* # of digits of key */
     i = 0;
  char key[3]; /* Allow maximum 3 appended table keys */
  i_key = table_array[table_list[table_cursor]].table_key;
  if (0 <= i_key && i_key <= 9)
```

```
  {
        key[0] = i_key + 48;  /* int 0 converts to char 0 */
        key_no = 1;
  }
 if (10 <= i_key && i_key <= 99)
  {
        key[1] = (i_key / 10) + 48;  /* 1st append key */
        key[0] = (i_key % 10) + 48;  /* 2nd append key */
        key_no = 2;
  }
 if (100 <= i_key && i_key <= 999)
  {
        key[2] = (i_key / 100) + 48;        /* 1st append key */
        key[1] = ((i_key % 100) / 10) + 48;  /* 2nd append key */
        key[0] = (i_key % 10) + 48;         /* 3rd append key */
        key_no = 3;
  }
 index = strlen(media_name);
 if ((index + key_no) >= 12)  /* Maximum length of att_name */
  {
        media_name[12] = '\0';  /* Assign '\0' to the end of string */
        for (i = 0; i < key_no; i++)
         media_name[index - (i + 1)] = key[i];
  }
 else
  {
        media_name[index + key_no] = media_name[index]; /* Move '\0' to end*/
        for (i = 0; i < key_no; i++)
         media_name[(index + key_no) - (i + 1)] = key[i];
  }  /* End of if else */
} /* End of get_media_name() */
/*****************************************************************************/
/* Translate SQL statement to create a MEDIA relation                        */
/*****************************************************************************/
void ql_create_media_table()
{
  int i = 0;
  for (i = 0; i < act_media_count; i++)
  {
   strcpy(media_name, att_array[act_media_list[i]].att_name);
   get_media_name();
   printf(" create table %12s (", media_name);
   strcpy(data_type, att_array[act_media_list[i]].data_type);
   if (strcmp(data_type, "image") == 0)
        {
          printf("i_id integer,\n                    ");
          printf("f_id c64,\n                  ");
          printf("descrp vchar500,\n                      ");
          printf("height integer,\n                     ");
          printf("width integer,\n                     ");
         printf("depth integer);\n\n");
        }
    else
        {
          printf("s_id integer,\n                    ");
```

```
            printf("f_id c64,\n                    ");
            printf("descrp vchar500,\n                    ");
            printf("size integer,\n                    ");
            printf("samp_rate integer,\n                    ");
            printf("encoding integer,\n                    ");
            printf("duration float,\n                    ");
            printf("resolution integer);\n\n");
          } /* End of if else */
/***************CREATE MEDIA TABLE IN INGRES START HERE*********************/
/***************THE INGRES FUNCTION CALLS WRITE MANULLY*********************/
/* # line 1046 "db.sc" */    /* create table */
 {
  printf("\nCREATING MEDIA TABLE NOW. PLEASE WAIT!!\n");
  IIsqInit(&sqlca);
  IIwritedb("create ");
  IIwritedb(media_name);
  IIwritedb("(");
  if (strcmp(data_type, "image") == 0)
    {
        IIwritedb("i_id=i4,f_id=c64,descrp=text(500),"); /* vchar(500) */
        IIwritedb("height=i4,width=i4,depth=i4)");
        printf("\nCREATE AN IMAGE TABLE COMPLETE!!\n");
    }
   else
    {
        IIwritedb("s_id=i4,f_id=c64,descrp=text(500),"); /* vchar(500) */
        IIwritedb("size=i4,samp_rate=i4,encoding=i4,");
        IIwritedb("duration=f4,resolution=i4)");
        printf("\nCREATE A SOUND TABLE COMPLETE!!\n");
    } /* End of if else */
  IIsqSync(0,&sqlca);
 }
/* # line 1068 "db.sc" */    /* host code */
/***************CREATE MEDIA TABLE IN INGRES STOP HERE**********************/
    while ((c = getchar()) != '\n')
      ;
  } /* End of for loop */
} /* End of ql_create_media_table() */
/*****************************************************************************/
/* Translate SQL statement to create a STANDARD relation                     */
/*****************************************************************************/
void ql_create_table()
{
  int i = 0,
     entry = 0,
     count = 0;
  act_media_count = 0;
  entry = table_array[table_list[table_cursor]].att_entry;
  count = table_array[table_list[table_cursor]].att_count;
  printf("\nSQL statement::\n");
  printf(" create table %12s (",
                        table_array[table_list[table_cursor]].table_name);
  for (i = 1; i < count; i++)
    {
        printf("%s ", att_array[entry].att_name);
```

```
        strcpy(data_type, att_array[entry].data_type);
        if ((strcmp(data_type, "image") == 0) ||
                              (strcmp(data_type, "sound") == 0))
          {
            printf("integer,\n");
            act_media_list[act_media_count] = entry;
            act_media_count ++;
          }
        else
          printf("%s,\n", att_array[entry].data_type);
        printf("                              ");
        entry = att_array[entry].next_index;
    } /* End of for loop i */
  printf("%s ",att_array[entry].att_name);
  strcpy(data_type, att_array[entry].data_type);
  if ((strcmp(data_type, "image") == 0) ||
                        (strcmp(data_type, "sound") == 0))
    {
        printf("integer);\n\n");
        act_media_list[act_media_count] = entry;
        act_media_count ++;
    }
  else
    printf("%s);\n\n", att_array[entry].data_type);
/****************CREATE STD TABLE IN INGRES START HERE**********************/
/*************THE INGRES FUNCTION CALLS WRITTEN MANULLY*******************/
  entry = table_array[table_list[table_cursor]].att_entry;
  count = table_array[table_list[table_cursor]].att_count;
/* # line 1120 "db.sc" */    /* create table */
  {
   printf("\nCREATING STD TABLE NOW. PLEASE WAIT!!\n");
   IIsqInit(&sqlca);
   IIwritedb("create ");
   IIwritedb(table_array[table_list[table_cursor]].table_name);
   IIwritedb("(");
    for (i = 1; i < count; i++)
      {
        IIwritedb(att_array[entry].att_name);
        IIwritedb("=");
        strcpy(data_type, att_array[entry].data_type);
        if ((strcmp(data_type, "image") == 0) ||
                          (strcmp(data_type, "sound") == 0) ||
                                (strcmp(data_type, "integer") == 0))
          IIwritedb('i4,");
        else
          if (strcmp(data_type, "float") == 0)
            IIwritedb("f4,");
          else
            {                                  /* char data_type */
             IIwritedb(att_array[entry].data_type);
             IIwritedb(",");
            }
        entry = att_array[entry].next_index;
    } /* End of for loop i */
   IIwritedb(att_array[entry].att_name);
```

```
  IIwritedb("=");
  strcpy(data_type, att_array[entry].data_type);
  if ((strcmp(data_type, "image") == 0) ||
                    (strcmp(data_type, "sound") == 0) ||
                                  (strcmp(data_type, "integer") == 0))
    IIwritedb("i4)");                    /* Integer type */
   else
    if (strcmp(data_type, "float") == 0)      /* Float type   */
        IIwritedb("f4)");
    else
        {                                /* char 20 type */
         IIwritedb(att_array[entry].data_type);
         IIwritedb(")");
        }
  IIsqSync(0,&sqlca);
  printf("\nCREATE A STD TABLE COMPLETE!!\n");
 }
/* # line 1164 "db.sc" */    /* host code */
/***************CREATE STD TABLE IN INGRES STOP HERE**********************/
  while ((c = getchar()) != '\n')
  ;
  if (act_media_count > 0)
   ql_create_media_table();
} /* End of ql_create_table() *
/*****************************************************************************/
/************************* Start for INSERTION ***************************/
/*****************************************************************************/
/*****************************************************************************/
/* Print out the table catalog information on screen                 */
/*****************************************************************************/
void print_all_table()
{
  int i = 0;
  printf("\t**Table Name**\n");
  for (i = 0; i < table_count; i++)
   {
    printf("\t  %s\n",table_array[table_list[i]].table_name);
    if ((i % 15) == 14)
        {
         printf("\n*RETURN TO CONTINUE*\n");
         while ((c = getchar()) != '\n')
         ;
         printf("\t**Table Name**\n");
        }
   } /* End of for loop */
} /* End of print_all_table() */
/*****************************************************************************/
/* Get a INTEGER value of a standard attribute from the user input       */
/*****************************************************************************/
void get_int_value()
{
 char stuff[3]; /* To provide a dummy var for '\n' when user enter '?' */
 i_value[i_index] = 0;
 scanf("%d", &i_value[i_index]);
 if (i_value[i_index] == 0)        /* ? or 0 entered */
```

```
   {
    i_value[i_index] = 0;          /* if 0 entered still 0 */
    stuff[0] = '\0';
    gets(stuff); /* To let next gets() work when ? entered in scanf() */
   }
  else
   getchar(); /* Add after scanf() to let next gets() work properly */
  att_array[att_cursor].value_entry = i_index;
  i_index = (i_index + 1) % 20;
} /* End of get_int_value() */
/*****************************************************************************/
/* Get a FLOAT value of a standard attribute from the user input          */
/*****************************************************************************/
void get_float_value()
{
 char stuff[3]; /* To provide a dummy var for '\n' when user enter '?' */
 f_value[f_index] = 0.0;
 scanf("%f", &f_value[f_index]);
 if (f_value[f_index] == 0.0)       /* ? or 0 entered */
   {
    f_value[f_index] = 0.0;         /* if 0 entered still 0.0 */
    stuff[0] = '\0';
    gets(stuff); /* To let next gets() work when ? entered in scanf() */
   }
  else
   getchar(); /* Add after scanf() to let next gets() work properly */
  att_array[att_cursor].value_entry = f_index;
  f_index = (f_index + 1) % 20;
} /* End of get_float_value() */
/*****************************************************************************/
/* Get a STRING value of a standard attribute from the user input         */
/*****************************************************************************/
void get_c20_value()
{
  int over_length = TRUE; /* Initialize to true */
  char c_temp[60]; /* Temp var for read in, 60 to avoid bus error */
  while (over_length)
    {
        c_temp[0] = '\0';
        gets(c_temp);
        if (strlen(c_temp) >= 21)
          {
           printf("\nSorry!! Value OVER 20 characters!");
           putchar('\007');
           printf("\nPlease Enter <<%s>> Value ( ? if unknow):: ",
                                                          data_type);
          }
        else
          {
           over_length = FALSE;
           strcpy(c_value[c_index], c_temp);
           if (strcmp(c_value[c_index], "?") == 0)
             strcpy(c_value[c_index], " "); /* Assign blank as null */
           att_array[att_cursor].value_entry = c_index;
           c_index = (c_index + 1) % 20;
```

```
        } /* End of if else */
    } /* End of while (over_length) */
} /* End of get_c20_value() */
/*****************************************************************************/
/* Get the description of a MEDIA attribute from the user input          */
/*****************************************************************************/
void get_descrp()
{
  char phrase[MAX_PHRASE+20]; /* Maximum length of a phrase is 127 */
  int phrase_len = 0,         /* Declared 20 char more to avoid the*/
     descrp_len = 0;          /* bus error!                        */
  int stop_input = FALSE;
  descrp[0] = '\0';
  printf("\nPlease Enter Description:");
  printf("\nNOTE: One phrase per line. End with an empty line::\n");
  while (!stop_input)
   {
   phrase[0] = '\0';
   gets(phrase);
   phrase_len = strlen(phrase);
   if (phrase_len >= 1)
     {
     if (phrase_len >= MAX_PHRASE) /*Need end with \n & \0 in one phrase*/
          {
          printf("\nThe phrase OVER %d characters!", (MAX_PHRASE - 1));
          printf("\nInvalid input!! TRY AGAIN!!\n");
          putchar('\007');
          }
         else
          {
          phrase[phrase_len] = '\n';
          phrase[phrase_len + 1] = '\0';
          if (phrase_len > 1)
           {
           if ((descrp_len + phrase_len + 1) >= (MAX_DESCRP + 1))
            {
             stop_input = TRUE;
            printf("\nThe last phrase extended beyond the maximum %d ",
                                                                  MAX_DESCRP);
            printf("\ncharacters in description. It has been canceled!\n");
             putchar('\007');
             while ((c = getchar()) != '\n')
              ;
             }
            else
             {
             strcat(descrp, phrase);
             descrp_len = descrp_len + phrase_len + 1;
             } /* End of if else */
           }; /* End of if (phrase_len > 1) */
          } /* End of if else (phrase_len >= MAX_PHRASE) */
        } /* End of if (phrase_len >= 1) */
      else /* Empty string input */
          {
          if (descrp_len == 0)
```

```
            {
             printf("\nSorry! Empty string is NOT allowed!!\n");
             putchar('\007');
            }
           else
            stop_input = TRUE;
          } /* End of if else */
   } /* End of while (!stop_input) */
} /* End of get_descrp() */
/*****************************************************************************/
/* Display the IMAGE by passing pixels and colormap from the caller.      */
/* It open another process in SUN environment to display the image on the  */
/* screen. It might be able to quit the image automatically before display */
/* the next image.                                                          */
/*****************************************************************************/
show_image(pixels, colormap)
struct pixrect *pixels;
colormap_t *colormap;
{
 char answer;
 int i, error, pid;
 Frame frame;
 Canvas canvas;
 Pixwin *pw;
 pid = fork ();
 if (pid < 0)
   {
    printf ("Starting display process failed\n\n");
    return (-1);
   }
 if (pid > 0) {
          return (pid);
          }
 if (colormap == NULL)
   {
    printf ("Cannot show it - no colormap.\n\n");
    exit (1);
   }
 frame = window_create (NULL, FRAME,
                        FRAME_LABEL,        "IMAGE",
                        FRAME_NO_CONFIRM,   TRUE,
                        WIN_WIDTH,          pixels->pr_size.x + 20,
                        WIN_HEIGHT,         pixels->pr_size.y + 50,
                        WIN_ERROR_MSG,      "Cannot create window.",
                        0);
 if (frame == NULL)
   {
    printf ("Cannot create frame\n\n");
    exit (1);
   };
 canvas = window_create (frame, CANVAS,
                         WIN_WIDTH,         pixels->pr_size.x,
                         WIN_HEIGHT,        pixels->pr_size.y,
                         0);
 if (canvas == NULL)
```

```
  {
   printf ("Cannot create canvas\n\n");
   exit (1);
  };
 pw = canvas_pixwin (canvas);
 if (pw == NULL)
  {
   printf("pixwin is NULL\n\n");
   exit (1);
  }
 window_fit (frame);
 if (colormap->type == RMT_EQUAL_RGB && colormap->length > 0)
  {
   pw_setcmsname(pw, "photo");
   if (error = pw_putcolormap(pw, 0, colormap->length,
                                          colormap->map[0],
                                          colormap->map[1],
                                          colormap->map[2]))
    {
        printf ("Cannot load colormap.\n");
        printf ("error code = %d\n", error);
        printf ("type = %d\nlength = %d\n", colormap->type,
                                              colormap->length);
        exit (1);
    };
  }
 else
  {
   printf ("Cannot show photo - colormap not appropriate.\n\n");
   exit (1);
  }
 if (pw_write (pw, 0, 0, pixels->pr_size.x, pixels->pr_size.y,
                                                  PIX_SRC, pixels, 0, 0))
  printf ("Cannot display image on screen.\n\n");
 else
  window_main_loop(frame);
 window_destroy(frame);
 pr_destroy(pixels);
 exit (0);
 return (0);
} /* End of show_image(pixels, colormap) */
/*****************************************************************************/
/* Get a IMAGE value of a media attribute from the user input          */
/*****************************************************************************/
void get_image_value()
{
  STR_path file_name;
  STR_descrp nothing;
  char temp_file[100]; /* Declare more to avoid bus error */
  int height = 0,
     width = 0,
     depth = 0;
  struct pixrect *pr;
  colormap_t cm;
  int show_pid, wait_pid;
```

```
union wait status;
int over_length = TRUE;  /* Initialize to true */
cm.type = RMT_NONE;      /* this is absolutely necessary! Otherwise     */
cm.length = 0;          /* pr_load_colormap might not allocate storage */
cm.map[0] = NULL;        /* for the colormap, if the garbage found in   */
cm.map[1] = NULL;        /* the cm structure seems to make sense. The   */
cm.map[2] = NULL;        /* result, of course, is segmentation fault.   */
img_record[img_index].i_id = att_array[att_cursor].media_id;
while (over_length)
 {
  printf("\nPlease Enter <<%s>> File Name!!", data_type);
  printf("\nNOTE: Enter The Full Path Name:: ( ? if unknow)\n");
  temp_file[0] = '\0';
  gets(temp_file);
  if (strlen(temp_file) >= (MAX_PATH +1))
   {
        printf("\nSorry!! PATH_NAME OVER %d characters! TRY AGAIN!!\n",
                                                                    MAX_PATH);
        putchar('\007');
   }
   else
   {
        strcpy(file_name, temp_file);
        if (strcmp(file_name, "?") == 0)
         {
          over_length = FALSE;
          strcpy(img_record[img_index].f_id, " ");
          strcpy(img_record[img_index].descrp, " ");
          img_record[img_index].height = height;
          img_record[img_index].width = width;
          img_record[img_index].depth = depth;
         }
        else
         {
          if ((img_file=fopen(file_name, "r")) == NULL)
           {
            printf("\n%s", file_name);
            printf("\nThe File cannot be opened! Try Again!!\n");
            putchar('\007');
           }
          else {
            pr = pr_load(img_file, &cm);  /* Get registration data */
         ISimage_from_pixrect(pr, &cm, file_name, nothing);
            if (pr == NULL)
                 {
                  printf("\n%s", file_name);
                  printf("\nThe File does not contain a proper image!");
                  printf("\nThe image must be in Sun Raster format!");
                  printf(" Try Again!!\n");
                  putchar('\007');
                 }
             else {
                  over_length = FALSE;
                  strcpy(img_record[img_index].f_id, file_name);
                  printf("\nDisplay the image before enter the description?");
```

```
                    printf(" (y/n):: ");
                    if (yes_no_answer() == 'y')
                      show_image(pr, &cm);
                    img_record[img_index].height = pr->pr_size.y;
                    img_record[img_index].width = pr->pr_size.x;
                    img_record[img_index].depth = pr->pr_depth;
                   } /* End of if else */
                } /* End of if else */
               fclose(img_file);
              } /* End of if else */
     } /* End of if else */
   } /* End of while (over_length) */
} /* End of get_image_value() */
/*****************************************************************************/
/* Play the SOUND before enter description                                   */
/*****************************************************************************/
void play_snd()
{
  char display = 'y';
  while (display == 'y')
    {
     play_sound(snd_record[snd_index].f_id);
          printf("\nPlaying sound.....");
          while (getchar() != '\n')
          ;
          printf("\nPlay one more time? (y/n)::");
          display = yes_no_answer();
    };
} /* End of play_snd() */
/*****************************************************************************/
/* Get a SOUND value of a media attribute from the user input                */
/*****************************************************************************/
void get_sound_value()
{
  STR_path file_name;
  char temp_file[100]; /* Declare more to avoid bus error */
  int size = 0,
     samp_rate = 0,
     encoding = 0,
     resolution = 0;
  float duration = 0.0;
  int over_length = TRUE; /* Initialize to true */
  snd_record[snd_index].s_id = att_array[att_cursor].media_id;
  while (over_length)
   {
    printf("\nPlease Enter <<%s>> File Name!!", data_type);
    printf("\nNOTE: Enter The Full Path Name:: ( ? if unknow)\n");
    temp_file[0] = '\0';
    gets(temp_file);
    if (strlen(temp_file) >= (MAX_PATH +1))
     {
          printf("\nSorry!! PATH_NAME OVER %d characters! TRY AGAIN!!\n",
                                                              MAX_PATH);
          putchar('\007');
     }
```

```
    else
     {
          strcpy(file_name, temp_file);
          if (strcmp(file_name, "?") == 0)
           {
            over_length = FALSE;
            strcpy(snd_record[snd_index].f_id, " ");
            strcpy(snd_record[snd_index].descrp, " ");
            snd_record[snd_index].size = size;
            snd_record[snd_index].samp_rate = samp_rate;
            snd_record[snd_index].encoding = encoding;
            snd_record[snd_index].duration = duration;
            snd_record[snd_index].resolution = resolution;
           }
          else
           {
           if ((snd_file = fopen(file_name, "r")) == NULL)
            {
             printf("\n%s", file_name);
             printf("\nThe File cannot be opened! Try Again!!\n");
             putchar('\007');
            }
           else
            {
          s_hdr.sfname[0] = '\0';
             snd_load(file_name);    /*Get registra from sound text file*/
             if (strlen(s_hdr.sfname) != 12) /* sfname must 12 chars as */
                  {                          /* a test of sound file    */
                   printf("\n%s", file_name);
                   printf("\nThe File does not contain a proper sound!");
                   printf(" Try Again!!\n");
                   putchar('\007');
                  }
              else                  /* i.e. Valid input */
                  {
                   over_length = FALSE;
           strcpy(snd_record[snd_index].f_id, s_hdr.sfname);
                   printf("\nPlay the sound before enter the description?");
                   printf(" (y/n)::");
                   if (yes_no_answer() == 'y')
                    play_snd();
           snd_record[snd_index].size = s_hdr.s_size;
           snd_record[snd_index].samp_rate = s_hdr.s_samplrate;
           snd_record[snd_index].encoding = s_hdr.s_encoding;
           snd_record[snd_index].duration = s_hdr.s_duration;
           snd_record[snd_index].resolution = s_hdr.s_resolution;
                  } /* End of if else */
            } /* End of if else */
           fclose(snd_file);
          } /* End of if else */
     } /* End of if else */
   } /* End of while (over_length) */
} /* End of get_sound_value() */
/*************************************************************************/
/* Get a value of a standard attribute from the user input               */
```

```
/*****************************************************************************/
void get_std_value()
{
  printf("\nTable Name:: %s\nAtt Name  :: %s\nData Type :: %s",
                    table_array[table_list[table_cursor]].table_name,
                                    att_array[att_cursor].att_name,
                                    att_array[att_cursor].data_type);
  printf("\nPlease Enter <<%s>> Value ( ? if unknow):: ", data_type);
  if (strcmp(data_type, "integer") == 0)
    get_int_value();                    /* Integer data type */
  else
   if (strcmp(data_type, "float") == 0)
     get_float_value();                 /* Float data tupe */
   else
     get_c20_value();                   /* String c20 data tupe */
} /* End of get_std_value() */
/*****************************************************************************/
/* Get a value of a media attribute from the user input                      */
/*****************************************************************************/
void get_media_value()
{
  printf("\nTable Name:: %s\nAtt Name  :: %s\nData Type :: %s",
                    table_array[table_list[table_cursor]].table_name,
                                    att_array[att_cursor].att_name,
                                    att_array[att_cursor].data_type);
  if (strcmp(data_type, "image") == 0)
    {
         img_value[img_index] = att_array[att_cursor].media_id;
         att_array[att_cursor].value_entry = img_index;
         get_image_value();             /* Image data type */
     if (strcmp(img_record[img_index].f_id, " ") != 0)
        {
         printf("\nEnter the description? (y/n):: ");
         if (yes_no_answer() == 'y')
           get_descrp();
          else
           strcpy(descrp, " ");
        strcpy(img_record[img_index].descrp, descrp);
        }
         att_array[att_cursor].media_id++;
         img_index = (img_index + 1) % 20;
    }
  else
    {
         snd_value[snd_index] = att_array[att_cursor].media_id;
         att_array[att_cursor].value_entry = snd_index;
         get_sound_value();             /* Sound data tupe */
     if (strcmp(snd_record[snd_index].f_id, " ") != 0)
        {
         printf("\nEnter the description? (y/n):: ");
         if (yes_no_answer() == 'y')
           get_descrp();
          else
           strcpy(descrp, " ");
        strcpy(snd_record[snd_index].descrp, descrp);
```

```
        }
          att_array[att_cursor].media_id++;
          snd_index = (snd_index + 1) % 20;
    } /* End of if else */
} /* End of get_media_value() */
/*****************************************************************************/
/* Get the values of a tuple from the user input. It begin loop at the 1st */
/* attribute until the last attribute entered                                */
/*****************************************************************************/
void get_tuple_value()
{
  int i = 0,
     count = 0;
  count = table_array[table_list[table_cursor]].att_count;
  att_cursor = table_array[table_list[table_cursor]].att_entry;
  act_media_count = 0;
  for (i = 0; i < count; i++) /* Loop to get value for each attribute */
    {
          strcpy(data_type, att_array[att_cursor].data_type);
          if ((strcmp(data_type, "image") == 0) ||
                                  (strcmp(data_type, "sound") == 0))
           {
            get_media_value();
            act_media_list[act_media_count] = att_cursor; /* Collect the */
             act_media_count++;                          /* media indices*/
           }
          else
           get_std_value();
          att_cursor = att_array[att_cursor].next_index;
    } /* End of for loop */
} /* End of get_tuple_value */
/*****************************************************************************/
/* Insert a tuple of one particular relation                                 */
/*****************************************************************************/
void insert_tuple()
{
  int table_found = FALSE; /* Initialize to false */
  while (!table_found)
  {
   printf("\nEnter table_name::(Maximum 12 characters); ( ? for HELP!)\n");
   table_name[0] = '\0';
   gets(table_name);
   if (strlen(table_name) >= 13) /* Over maximum name length */
     {
          printf("\nSorry!! Table Name OVER 12 characters!");
          putchar('\007');
     }
    else
     {
          if (strcmp(table_name, "?") == 0)
           print_all_table();
          else
           {
            strcpy(table_array[table_index].table_name, table_name);
            table_found = check_table_name();
```

```
            if (table_found)
            {
                table_cursor = table_entry;
                get_tuple_value();
               }
             else
                {
                 printf("\nSorry!! Table name: %s NOT found! TRY AGAIN!!",
                                        table_array[table_index].table_name);
                 putchar('\007');
                } /* End of if else */
            } /* End of if else */
     } /* End of if else */
  } /* End of while (!table_found) */
} /* End of insert_tuple() */
/*****************************************************************************/
/* Print out the value of current tuple which the user want to insert      */
/*****************************************************************************/
void print_tuple()
{
  int i = 0,
     count = 0,
     entry = 0;
  clr_scr();
  entry = table_array[table_list[table_cursor]].att_entry;
  count = table_array[table_list[table_cursor]].att_count;
  printf("\nTable Name:: %s\n",
                    table_array[table_list[table_cursor]].table_name);
  printf("\nOrder  Attribute Name\tData Type\tValue\n");
  for (i = 0; i < count; i++)
  {
   strcpy(data_type, att_array[entry].data_type);
   if (strcmp(data_type, "c20") == 0)
    printf(" %d  %13s\t%s\t\t'%s\'\n",(i+1), att_array[entry].att_name,
                                              att_array[entry].data_type,
                                    c_value[att_array[entry].value_entry]);
    else
    if (strcmp(data_type, "integer") == 0)
     printf(" %d  %13s\t%s\t\t%d\n",(i+1) , att_array[entry].att_name,
                                              att_array[entry].data_type,
                                    i_value[att_array[entry].value_entry]);
     else
     if (strcmp(data_type, "float") == 0)
          printf(" %d  %13s\t%s\t\t%f\n",(i+1) , att_array[entry].att_name,
                                              att_array[entry].data_type,
                                    f_value[att_array[entry].value_entry]);
     else
         if (strcmp(data_type, "image") == 0)
          {
           printf(" %d  %13s\t%s\t\t",(i+1) , att_array[entry].att_name,
                                              att_array[entry].data_type);
           if (strcmp(img_record[att_array[entry].value_entry].f_id, " ")
                                                                    == 0)
            printf("NO VALUE\n");
           else
```

```
            printf("HAS VALUE\n");
          }
          else
          {
            printf(" %d  %13s\t%s\t\t",(i+1) , att_array[entry].att_name,
                                              att_array[entry].data_type);
            if (strcmp(snd_record[att_array[entry].value_entry].f_id, " ")
                                                                    == 0)
              printf("NO VALUE\n");
            else
              printf("HAS VALUE\n");
          }
    entry = att_array[entry].next_index;
   } /* End of for loop i */
} /* End of print_tuple() */
/*****************************************************************************/
/* Print out the description of media attribute in current the tuple      */
/*****************************************************************************/
void print_media_tuple()
{
  int i = 0,
      entry;
  STR_name data_type;
  printf("\nMedia Description::\n");
  for (i = 0; i < act_media_count; i++)
   {
    printf("\nAtt_name  :: %s", att_array[act_media_list[i]].att_name);
    strcpy(data_type, att_array[act_media_list[i]].data_type);
    entry = att_array[act_media_list[i]].value_entry;
    if (strcmp(data_type, "image") == 0)
      {
           printf("\nFile_name  :: \'%s\'", img_record[entry].f_id);
           printf("\nDescription:: \n<<%s>>", img_record[entry].descrp);
      }
     else
      {
           printf("\nFile_name  :: \'%s\'", snd_record[entry].f_id);
           printf("\nDescription:: \n<<%s>>", snd_record[entry].descrp);
      }
     while ((c = getchar()) != '\n')
      ;
   } /* End of for loop */
} /* End of print_media_tuple() */
/*****************************************************************************/
/* Print out the value of current attribute                                  */
/*****************************************************************************/
void print_value()
{
  int entry;
  entry = att_array[att_cursor].value_entry;
  clr_scr();
  printf("\nTable Name:: %s",
                    table_array[table_list[table_cursor]].table_name);
  printf("\nAtt_Name  :: %s", att_array[att_cursor].att_name);
  printf("\nData Type :: %s", att_array[att_cursor].data_type);
```

```
  printf("\nValue     :: ");
  if (strcmp(data_type, "c20") == 0)
    printf("\'%s\'\n", c_value[entry]);
  else
   if (strcmp(data_type, "integer") == 0)
    printf("%d\n", i_value[entry]);
   else
    if (strcmp(data_type, "float") == 0)
          printf("%f\n", f_value[entry]);
     else
     if (strcmp(data_type, "image") == 0)
       {
            printf("\n\t==>File_name  :: \'%s\'", img_record[entry].f_id);
            printf("\n\t==>Description:: \n<<%s>>\n", img_record[entry].descrp);
       }
      else
       {
            printf("\n\t==>File_name  :: \'%s\'", snd_record[entry].f_id);
            printf("\n\t==>Description:: \n<<%s>>\n", snd_record[entry].descrp);
       }
} /* End of print_value() */
/*****************************************************************************/
/* Change the IMAGE values of current tuple which the user want to insert  */
/*****************************************************************************/
void change_img_value()
{
 int cursor;  /* Previous index of media record array */
 cursor = att_array[att_cursor].value_entry;
 img_value[img_index] = att_array[att_cursor].media_id;
 att_array[att_cursor].value_entry = img_index;
 printf("\nChange IMAGE file name? (y/n):: ");
 if (yes_no_answer() == 'y')
  get_image_value();  /* Image data type */
 else
   {
    img_record[img_index].i_id = att_array[att_cursor].media_id;
    strcpy(img_record[img_index].f_id, img_record[cursor].f_id);
    img_record[img_index].height = img_record[cursor].height;
    img_record[img_index].width = img_record[cursor].width;
    img_record[img_index].depth = img_record[cursor].depth;
   }
 printf("\nChange IMAGE description? (y/n):: ");
 if (yes_no_answer() == 'y')
   {
    get_descrp();
    strcpy(img_record[img_index].descrp, descrp);
   }
 else
  strcpy(img_record[img_index].descrp, img_record[cursor].descrp);
 att_array[att_cursor].media_id++;
 img_index = (img_index + 1) % 20;
} /* End of change_img_value() */
/*****************************************************************************/
/* Change the SOUND values of current tuple which the user want to insert  */
/*****************************************************************************/
```

```
void change_snd_value()
{
 int cursor; /* Previous index of media record array */
 cursor = att_array[att_cursor].value_entry;
 snd_value[snd_index] = att_array[att_cursor].media_id;
 att_array[att_cursor].value_entry = snd_index;
 printf("\nChange SOUND file name? (y/n):: ");
 if (yes_no_answer() == 'y')
  get_sound_value(); /* Sound data type */
 else
  {
   snd_record[snd_index].s_id = att_array[att_cursor].media_id;
   strcpy(snd_record[snd_index].f_id, snd_record[cursor].f_id);
   snd_record[snd_index].size = snd_record[cursor].size;
   snd_record[snd_index].samp_rate = snd_record[cursor].samp_rate;
   snd_record[snd_index].encoding = snd_record[cursor].encoding;
   snd_record[snd_index].duration = snd_record[cursor].duration;
   snd_record[snd_index].resolution = snd_record[cursor].resolution;
  }
 printf("\nChange SOUND description? (y/n):: ");
 if (yes_no_answer() == 'y')
  {
   get_descrp();
   strcpy(snd_record[snd_index].descrp, descrp);
  }
 else
  strcpy(snd_record[snd_index].descrp, snd_record[cursor].descrp);
 att_array[att_cursor].media_id++;
 snd_index = (snd_index + 1) % 20;
} /* End of change_snd_value() */
/*****************************************************************************/
/* Change the values of current tuple which the user want to insert      */
/*****************************************************************************/
void modify_tuple()
{
  int i = 0,
     count = 0,
     entry = 0,
     order = 0;
  char more_change = 'y';
  while (more_change == 'y')
  {
   print_tuple();
   printf("Select the order which you want to change its value::\n");
   printf("Any other key to cancel the operation!! Select::");
   scanf("%d", &order);
   getchar(); /* To let next gets() work properly */
   entry = table_array[table_list[table_cursor]].att_entry;
   count = table_array[table_list[table_cursor]].att_count;
   if (1 <= order && order <= count)
    {
        for (i = 1; i < order; i++)
         entry = att_array[entry].next_index;
        att_cursor = entry; /* Assign the current index of att_array */
        strcpy(data_type, att_array[att_cursor].data_type);
```

```
        print_value();
        printf("\nPlease Enter <<%s>> Value ( ? if unknow):: ", data_type);
        if (strcmp(data_type, "integer") == 0)
          get_int_value(); /* Integer data type */
        else
         if (strcmp(data_type, "float") == 0)
           get_float_value(); /* Float data tupe */
         else
          if (strcmp(data_type, "c20") == 0)
            get_c20_value(); /* String c20 data tupe */
          else
           if (strcmp(data_type, "image") == 0)
          change_img_value();
         else
          change_snd_value();
      print_value();
     }
    else
     {
          printf("\nSorry! You entered the wrong order!! Please redo again.\n");
          putchar('\007');
     } /* End of if else */
    printf("Any More Change? (y/n):: ");
    more_change = yes_no_answer();
   } /* End of while */
} /* End of modify_tuple() */
/*****************************************************************************/
/* Display the tuple before insertion                                        */
/*****************************************************************************/
void display_tuple()
{
  char modify = 'y';
  while (modify == 'y')
  {
    clr_scr();
    print_tuple();
    while ((c= getchar(`` != '\n')
     ;
    if (act_media_count >= 1)
          print_media_tuple();
    printf("\nAny change before insert? (y/n)::");
    modify = yes_no_answer();
    if (modify == 'y')
          modify_tuple();
  } /* End of while */
} /* End of display_info() */
/*****************************************************************************/
/* Connect to parser to generate the facts file. We put all media descrip- */
/* tion in one facts file "imagei_image_facts" at this time, it should be  */
/* separate later on.                                                        */
/*****************************************************************************/
int connect_parser(file_id, new_descrp, err_message)
STR_path *file_id;
STR_descrp *new_descrp;
char *err_message;
```

```
{
  STR_path nothing;
  STR_descrp empty_descrp;
  int ISerror = FALSE;
  empty_descrp[0] = '\0';
  nothing[0] = '\0';
  printf("\nConnect to PARSER, Please Wait.....\n");
  ISerror = ISreplace_description("image", "i_image", file_id, empty_descrp,
                    new_descrp, nothing, empty_descrp, err_message);
    /* HERE, ISfunction call, Connect to parser and generate the */
    /* facts file in "imagei_image_facts"                        */
  if (ISerror)
   return(ISerror);
  else
   return(FALSE);
} /* End of connect_parser() */
/*****************************************************************************/
/* Check the media description by connecting to parser                       */
/*****************************************************************************/
int check_media_descrp()
{
  int i = 0,
     entry;
  int error = FALSE;
  char *err_message;
  while (i < act_media_count && !error)
   {
    strcpy(data_type, att_array[act_media_list[i]].data_type);
    entry = att_array[act_media_list[i]].value_entry;
    if (strcmp(data_type, "image") == 0)
      {
       if (strcmp(img_record[entry].descrp, " ") != 0)
        error = connect_parser(img_record[entry].f_id,
                        img_record[entry].descrp, err_message);
      }
     else
      {
       if (strcmp(snd_record[entry].descrp, " ") != 0)
        error = connect_parser(snd_record[entry].f_id,
                        snd_record[entry].descrp, err_message);
      }
     i++;
   }
  if (error)
   {
    printf("\nThe description for media \'%s\' is NOT acceptable!",
                        att_array[act_media_list[i-1]].att_name);
    if (error == DESCR_WORD_ERR)
     printf("\nThe system cannot understand the word >>%s<<", err_message);
     else
      if (error == DESCR_STRUCTURE_ERR)
       printf("\nThe system cannot interpret the phase\n >>%s<<",
                                                        err_message);
       else
        printf("\nThe program error occur in prolog!\n");
```

```
    printf("\nPlease modify it. Thank you!");
    putchar('\007');
    while((c=getchar()) != '\n')
     ;
    return(TRUE);
   }
  else
   return(FALSE);
} /* End of check_media_descrp() */
/*****************************************************************************/
/* Translate SQL statement to insert a media tuple                           */
/*****************************************************************************/
void ql_insert_media_tuple()
{
  int i = 0,
     entry;
  for (i = 0; i < act_media_count; i++)
   {
    strcpy(media_name, att_array[act_media_list[i]].att_name);
    get_media_name();
    printf("  insert into %12s (", media_name);
    strcpy(data_type, att_array[act_media_list[i]].data_type);
    entry = att_array[act_media_list[i]].value_entry;
    if (strcmp(data_type, "image") == 0)
         {
           printf("i_id   ,\n                    ");
           printf("f_id   ,\n                    ");
           printf("descrp ,\n                    ");
           printf("height ,\n                    ");
           printf("width  ,\n                    ");
          printf("depth  )\n");
           printf("          values(");
           printf("  %d ,\n                ",
                                        img_record[entry].i_id);
           printf("\'%s\',\n             ",
                                       img_record[entry].f_id);
           printf("\'%s\',\n             ",
                                      img_record[entry].descrp);
           printf("  %d ,\n              ",
                                       img_record[entry].height);
           printf("  %d ,\n              ",
                                       img_record[entry].width);
          printf("  %d );\n\n", img_record[entry].depth);
         }
     else
         {
           printf("s_id     ,\n                  ");
           printf("f_id     ,\n                  ");
           printf("descrp   ,\n                  ");
           printf("size     ,\n                  ");
           printf("samp_rate,\n                  ");
           printf("encoding ,\n                  ");
           printf("duration ,\n                  ");
          printf("resolution)\n");
           printf("          values (");
```

```
            printf(" %d ,\n                ",
                                    snd_record[entry].s_id);
            printf("\'%s\',\n              ",
                                    snd_record[entry].f_id);
            printf("\'%s\',\n              ",
                                    snd_record[entry].descrp);
            printf(" %d ,\n                ",
                                    snd_record[entry].size);
            printf(" %d ,\n                ",
                                    snd_record[entry].samp_rate);
            printf(" %d ,\n                ",
                                    snd_record[entry].encoding);
            printf(" %f ,\n                ",
                                    snd_record[entry].duration);
           printf(" %d );\n\n", snd_record[entry].resolution);
         }
/***************INSERT MEDIA TUPLE IN INGRES START HERE*********************/
/***************THE INGRES FUNCTION CALLS WRITE MANULLY*********************/
/* # line 2100 "db.sc" */  /* insert */
  {
   printf("\nINSERTING MEDIA TUPLE NOW. PLEASE WAIT!!\n");
   IIsqInit(&sqlca);
   IIwritedb("append to ");
   IIwritedb(media_name);
   IIwritedb("(");
   if (strcmp(data_type, "image") == 0)
    {
    IIwritedb("i_id=");
    IIsetdom(1,30,4, &img_record[entry].i_id);
    IIwritedb(" ,f_id=");
    IIsetdom(1,32,0, img_record[entry].f_id);
    IIwritedb(" ,descrp=");
    IIsetdom(1,32,0, img_record[entry].descrp);
    IIwritedb(" ,height=");
    IIsetdom(1,30,4, &img_record[entry].height);
    IIwritedb(" ,width=");
    IIsetdom(1,30,4, &img_record[entry].width);
    IIwritedb(" ,depth=");
    IIsetdom(1,30,4, &img_record[entry].depth);
    IIwritedb(" )");
    printf("\nINSERT AN IMAGE TUPLE COMPLETE!!\n");
    }
   else
    {
    IIwritedb("s_id=");
    IIsetdom(1,30,4, &snd_record[entry].s_id);
    IIwritedb(" ,f_id=");
    IIsetdom(1,32,0, snd_record[entry].f_id);
    IIwritedb(" ,descrp=");
    IIsetdom(1,32,0, snd_record[entry].descrp);
    IIwritedb(" ,size=");
    IIsetdom(1,30,4, &snd_record[entry].size);
    IIwritedb(" ,samp_rate=");
    IIsetdom(1,30,4, &snd_record[entry].samp_rate);
    IIwritedb(" ,encoding=");
```

```
    IIsetdom(1,30,4, &snd_record[entry].encoding);
    IIwritedb(" ,duration=");
    IIsetdom(1,31,4, &snd_record[entry].duration);
    IIwritedb(" ,resolution=");
    IIsetdom(1,30,4, &snd_record[entry].resolution);
    IIwritedb(" )");
    printf("\nINSERT A SOUND TUPLE COMPLETE!!\n");
    }
  IIsqSync(3,&sqlca);
 }
/* # line 2147 "db.sc" */  /* insert */
/*****************INSERT MEDIA TUPLE IN INGRES STOP HERE*********************/
    while ((c = getchar()) != '\n')
     ;
   } /* End of for loop */
} /* End of ql_insert_media_tuple() */
/*****************************************************************************/
/* Translate SQL statement to insert a standard tuple                        */
/*****************************************************************************/
void ql_insert_tuple()
{
  int i = 0,
     count = 0,
     entry = 0;
  clr_scr();
  entry = table_array[table_list[table_cursor]].att_entry;
  count = table_array[table_list[table_cursor]].att_count;
  printf("\nSQL statement::\n");
  printf("  insert into %12s (",
                          table_array[table_list[table_cursor]].table_name);
  for (i = 1; i < count; i++)
   {
    printf("%12s,\n", att_array[entry].att_name);
     printf("                          ");
    entry = att_array[entry].next_index;
   }
  printf("%12s)\n", att_array[entry].att_name);
  printf("          values (");
  entry = table_array[table_list[table_cursor]].att_entry;
  for (i = 1; i < count; i++)
   {
    strcpy(data_type, att_array[entry].data_type);
    if (strcmp(data_type, "c20") == 0)
        printf("\'%s\',\n", c_value[att_array[entry].value_entry]);
    else
    if (strcmp(data_type, "integer") == 0)
        printf(" %d ,\n", i_value[att_array[entry].value_entry]);
    else
       if (strcmp(data_type, "float") == 0)
         printf(" %f ,\n", f_value[att_array[entry].value_entry]);
       else
        if (strcmp(data_type, "image") == 0)
          printf(" %d ,\n", img_value[att_array[entry].value_entry]);
        else
          printf(" %d ,\n", snd_value[att_array[entry].value_entry]);
```

```
    printf("                    ");
    entry = att_array[entry].next_index;
   }
  strcpy(data_type, att_array[entry].data_type);
  if (strcmp(data_type, "c20") == 0)
   printf("\'%s\');\n\n", c_value[att_array[entry].value_entry]);
  else
   if (strcmp(data_type, "integer") == 0)
    printf(" %d );\n\n", i_value[att_array[entry].value_entry]);
   else
    if (strcmp(data_type, "float") == 0)
     printf(" %f );\n\n", f_value[att_array[entry].value_entry]);
    else
     if (strcmp(data_type, "image") == 0)
          printf(" %d );\n\n", img_value[att_array[entry].value_entry]);
     el
          printf(" %d );\n\n", snd_value[att_array[entry].value_entry]);
/****************INSERT STD TUPLE IN INGRES START HERE*********************/
/***************THE INGRES FUNCTION CALLS WRITE MANULLY*********************/
 entry = table_array[table_list[table_cursor]].att_entry;
 count = table_array[table_list[table_cursor]].att_count;
/* # line 2213 "db.sc" */   /* insert */
 {
  printf("\nINSERTING STD TUPLE NOW. PLEASE WAIT!!\n");
  IIsqInit(&sqlca);
  IIwritedb("append to ");
  IIwritedb(table_array[table_list[table_cursor]].table_name);
  IIwritedb("(");
   for (i = 1; i < count; i++)
    {
    IIwritedb(att_array[entry].att_name);
    IIwritedb("=");
    strcpy(data_type, att_array[entry].data_type);
    if (strcmp(data_type, "c20") == 0)
          IIsetdom(1,32,0, c_value[att_array[entry].value_entry]);
     else
     if (strcmp(data_type, "integer") == 0)
          IIsetdom(1,30,4, &i_value[att_array[entry].value_entry]);
     else
         if (strcmp(data_type, "float") == 0)
           IIsetdom(1,31,4, &f_value[att_array[entry].value_entry]);
          else
           if (strcmp(data_type, "image") == 0)
            IIsetdom(1,30,4, &img_value[att_array[entry].value_entry]);
           else
             IIsetdom(1,30,4, &snd_value[att_array[entry].value_entry]);
    IIwritedb(" ,");
    entry = att_array[entry].next_index;
    }
  IIwritedb(att_array[entry].att_name);
  IIwritedb("=");
  strcpy(data_type, att_array[entry].data_type);
  if (strcmp(data_type, "c20") == 0)
   IIsetdom(1,32,0, c_value[att_array[entry].value_entry]);
  else
```

```
    if (strcmp(data_type, "integer") == 0)
     IIsetdom(1,30,4, &i_value[att_array[entry].value_entry]);
    else
     if (strcmp(data_type, "float") == 0)
          IIsetdom(1,31,4, &f_value[att_array[entry].value_entry]);
     else
     if (strcmp(data_type, "image") == 0)
          IIsetdom(1,30,4, &img_value[att_array[entry].value_entry]);
     else
          IIsetdom(1,30,4, &snd_value[att_array[entry].value_entry]);
   IIwritedb(" )");
   IIsqSync(3,&sqlca);
   printf("\nINSERT A STD TUPLE COMPLETE!\n");
  }
/* # line 2261 "db.sc" */   /* insert */
/*******************INSERT STD TUPLE IN INGRES STOP HERE*********************/
  while ((c = getchar()) != '\n')
   ;
  if (act_media_count >= 1)
   ql_insert_media_tuple();
} /* End of ql_insert_tuple() */
/*****************************************************************************/
/*              Begin for retrieve                                           */
/*****************************************************************************/
/* Procedure initialize the array to empty                                   */
/* Initialize all parameters used in the retrieve to null                    */
/*****************************************************************************/
void init()
{
        int i,j;
        icond=0;
        gcond=0;
        for (i=0;i<10;i++) {
                for (j=0;j < 13;j++) {
                        satt[i].t_name[j] =0;
                        satt[i].a_name[j] =0;
                        stab[i].t_name[j] =0;
                        att[i][j]=0;
                        tab[i][j]=0;
                        }
                for (j=0;j<100;j++) {
                        con[i][j]='0';
                }
        }
}
/*****************************************************************************/
/* This procedure get the table name, attribute name of that table           */
/* and then return the attribute type to the user                            */
/*****************************************************************************/
getatttype(tab_name,att_name,att_type)
STR_name tab_name;
STR_name att_name;
STR_name att_type;
{
  int i,j,k,found,count;
```

```
  found = 0;
  for (i=0;i < table_count;i++) {
        if (strcmp(table_array[i].table_name,tab_name)==0) {
          j = table_array[i].att_entry;
          count = table_array[i].att_count;
          i = 1000;
        }
    }
    for ( k=0;k < count;k++) {
        if (strcmp(att_array[j].att_name, att_name)==0) {
          strcpy(att_type,att_array[j].data_type);
/* For test only */
          printf("\n%s",att_array[j].att_name);
          printf("\t%s\n",att_type);
          found = 1;
          k = 1000;
        }
        j = att_array[j].next_index;
    }
}
/*****************************************************************************/
/* procedure search media attribute search for the media attribute in the */
/* Relation and return m_att to caller                                       */
/*****************************************************************************/
void search_media_att (m_att)
STR_name m_att;
{
int j;
   for (j=0;j<numcon;j++) {
        if (contype[j]==1) {
                strcpy(m_att,att[j]);
      }
        if (contype[j]==2) {
                strcpy(m_att,att[j]);
        }
   }
}
/*****************************************************************************/
/* procedure to process the image condition                                  */
/* put the result in the media tale [number condition] for process later  */
/*****************************************************************************/
void process_icon2(query_phrase,number)
char query_phrase[DESCRLEN+1];
int number;
{
 int id;
 char answer, repeat, yes_no_answer (),con_number;
 int i, query_err, query_len, in_len, f_flag,found;
 struct pixrect *pr;
 colormap_t cm;
 char descr[DESCRLEN+1];
 int show_pid, wait_pid;
 union wait status;
 int imageno;
 printf ("\nEntering RETRIEVE ...\n");
```

```
 cm.type = RMT_NONE;
 cm.length = 0;
 cm.map[0] = NULL;
 cm.map[1] = NULL;
 cm.map[2] = NULL;
 /* this is absolutely necessary!!!! Otherwise pr_load_colormap might
   not allocate storage for the colormap, if the garbage found in
   the cm structure seems to make sense. The result, of course, is
   segmentation fault. This bug was very hard to find. */
 {
/* # line 193 "p2.sc" */    /* create table */
 {
  IIsqInit((char *)0);
  IIwritedb("create m");
  IIwritedb("(i_id=i4)");
  IIsqSync(0,(char *)0);
 }
/* # line 194 "p2.sc" */    /* host code */
     printf("The query description now is:\n>>%s<<\n\n",query_phrase);
         printf ("Searching .....\n");
   /* exec sql declare c1 cursor for
         select i_id, PIXRECT (i_image), COLORMAP (i_image),
                DESCRIPTION (i_image)
          from  emp_img1
         where SHOWS (i_image, query_phrase);
         The statement is deleted by the preprocessor.
     However, the output functions and the selection conditions
         associated with the cursor c1 will be used later.
         The following declarations are generated: */
     {
         int ISerrorc1;
     char ISerrmcc1[ERRMLEN+1];
 char ISfnc1[FILENAMELEN + 1];
 char ISdescrc1[DESCRLEN + 1];
     sqlca.sqlcode = 0;
     ISerrmcc1[0] = '\0';
   /* exec sql open c1; */
   /* exec sql whenever not found go to closec1; */
   /* translated by preprocessor into: */
     if ( ISerrorc1 = ISshows_open("image","i_image",ISfnc1,query_phrase,ISerrmcc1) )
      {
       sqlca.sqlcode = ISerrorc1;
           if ( sqlca.sqlcode == QUERY_WORD_ERR ||
                   sqlca.sqlcode == QUERY_STRUCTURE_ERR )
        strcpy(sqlca.sqlerrm.sqlerrmc,ISerrmcc1);
      }
   /* end of preprocessor output for open c1 */
     if ( !sqlca.sqlcode )
      {
       f_flag = 0;
       for (;;)
       {
       /* exec sql fetch c1
            into :imageno, :pr, :cm, :descr;
            This is translated by the preprocessor into: */
```

```
        if ( ISerrorc1 = ISshows_fetch("image","i_image",ISfnc1,query_phrase,ISerrmcc1) )
           sqlca.sqlcode = ISerrorc1;
         if ( sqlca.sqlcode == NOT_FOUND )
            goto closec1;
          f_flag = 1;
         if ( !sqlca.sqlcode )
          {
/* # line 653 "p1.sc" */    /* select */
 strcpy (table_array[table_index].table_name, tab[number]);
 found = check_table_name();
 table_cursor = table_entry;
 strcpy(media_name,att[number]);
 get_media_name();
 printf("%s",media_name);
 {
  IIsqInit(&sqlca);
  IIwritedb("retrieve(imageno=");
  IIwritedb(media_name);
  IIwritedb(".i_id,ISdescrc1=");
  IIwritedb(media_name);
  IIwritedb(".descrp)w");
  IIwritedb("here ");
  IIwritedb(media_name);
  IIwritedb(".f_id=");
  IIsetdom(1,32,0,ISfnc1);
  IIwritedb(" ");
  IIsqRinit(&sqlca);
  if (IIerrtest() == 0) {
   if (IInextget() != 0) {
    IIretdom(1,30,4,&imageno);
    IIretdom(1,32,0,ISdescrc1);
   } /* IInextget */
   IIsqFlush(&sqlca);
  } /* IIerrtest */
 }
/* # line 657 "p1.sc" */    /* host code */
           if (!sqlca.sqlcode)
            {
               if (!(ISerrorc1 = ISpixrect (ISfnc1, ISdescrc1, &pr)))
                 if (!(ISerrorc1 = IScolormap (ISfnc1, ISdescrc1, &cm)))
                   ISerrorc1 = ISdescription (ISfnc1, ISdescrc1, descr);
              sqlca.sqlcode = ISerrorc1;
               }
             else
              sqlca.sqlcode = PROGRAM_ERR;
           }
           /* end of preprocessor output for fetch c1 */
             if (sqlca.sqlcode)
                goto closec1;
              id = imageno;
/* # line 270 "p2.sc" */    /* insert */
 {
  IIsqInit((char *)0);
  IIwritedb("append to m");
  IIwritedb("(i_id=");
```

```
  IIsetdom(1,30,4,&id);
  IIwritedb(" )");
  IIsqSync(3,(char *)0);
 }
/* # line 272 "p2.sc" */    /* host code */
        } /* end for loop of cursor c1 */
       closec1:
           /* exec sql close c1; */
           /* translated by the preprocessor into: */
       sqlca.sqlcode = ISshows_close("image","i_image",ISfnc1,query_phrase,ISerrmcc1);
/* # line 693 "p1.sc" */    /* host code */
          }  /* end of successful open c1; correct query description  */
          }  /* end of preprocessor declaration block */
     if ( sqlca.sqlcode == QUERY_WORD_ERR )
      {
       printf("The system cannot understand the word >>%s<<\n",sqlca.sqlerrm.sqlerrmc);
        query_err = 1;
      }
     if ( sqlca.sqlcode == QUERY_STRUCTURE_ERR )
      {
       printf("The system cannot interpret the phrase\n>>\n%s<<\n",sqlca.sqlerrm.sqlerrmc);
        query_err = 1;
      }
     if ( query_err )
      {
      }
    }
    if ( !f_flag )
      printf("There are no media matching that query description.\n");
    if ( sqlca.sqlcode )
      printf("An error has occured while accessing the database\n\
sql error code: %d\n", sqlca.sqlcode);
   clr_scr();
}  /* end of retrieve_photo () */
/*************************************************************************/
/* present photo the the user present number and description too  */
/*************************************************************************/
present_photo (number, pixels, colormap, description)
int number;
struct pixrect *pixels;
colormap_t *colormap;
char *description;
{
 char answer, yes_no_answer ();
 int i, error, pid;
 Frame frame;
 Canvas canvas;
 Pixwin *pw;
 printf ("\nThe following photo has been found:\n\n");
 printf ("Number: %d\n", number);
 printf ("Description:\n>>%s<<\n\n", description);
 printf ("Do you want to see the photo? ");
 answer = yes_no_answer ();
 if (answer == 'n')
  return (0);
```

```
else {
 pid = fork ();
 if (pid < 0) {
  printf ("Starting display process failed\n\n");
  return (-1);
 }
 if (pid > 0)  /* this is parent process */
  return (pid);
 if (colormap == NULL) {
  printf ("Cannot show it - no colormap.\n\n");
  exit (1);
 }
 frame = window_create (NULL, FRAME,
                        FRAME_LABEL,        "IMAGE",
                        FRAME_NO_CONFIRM,   TRUE,
                        WIN_WIDTH,          pixels->pr_size.x + 20,
                        WIN_HEIGHT,         pixels->pr_size.y + 50,
                        WIN_ERROR_MSG,      "Cannot create window.",
                        0);
 if (frame == NULL) {
  printf ("Cannot create frame\n\n");
  exit (1);
 }
 canvas = window_create (frame, CANVAS,
                         WIN_WIDTH,         pixels->pr_size.x,
                         WIN_HEIGHT,        pixels->pr_size.y,
                         0);
 if (canvas == NULL) {
  printf ("Cannot create canvas\n\n");
  exit (1);
 }
 pw = canvas_pixwin (canvas);
 if (pw == NULL) {
  printf ("pixwin is NULL\n\n");
  exit (1);
 }
 window_fit (frame);
 if (colormap->type == RMT_EQUAL_RGB
       && colormap->length > 0) {
  pw_setcmsname (pw, "photo");
  if (error = pw_putcolormap (pw, 0, colormap->length,
                colormap->map[0], colormap->map[1], colormap->map[2])) {
       printf ("Cannot load colormap.\n");
       printf ("error code = %d\n", error);
       printf ("type = %d\nlength = %d\n", colormap->type, colormap->length);
  /* for (i = 0; i < colormap->length; i++) {
        printf (" %x %x %x\n", *(colormap->map[0] + i),
                *(colormap->map[1] + i), *(colormap->map[2] + i));
       } */
       exit (1);
  }
 }
 else {
  printf ("Cannot show photo - colormap not appropriate.\n\n");
  exit (1);
```

```
    }
    if (pw_write (pw, 0, 0, pixels->pr_size.x, pixels->pr_size.y,
                  PIX_SRC,
                  pixels, 0, 0))
      printf ("Cannot write image to screen.\n\n");
    else
      window_main_loop (frame);
    window_destroy (frame);
    exit (0);
  }  /* of (answer = 'y'), showing the photo */
  return (0);
}
/*************************************************************************/
/* This procedure search through the media relation and get the   */
/* file name that match with the result table and send to the     */
/* present photo procedure                                        */
/*************************************************************************/
display_photo (imageno,tupleno)
int imageno;
int tupleno;
{
  char answer, repeat, yes_no_answer ();
  char query_phrase[DESCRLEN+1],
       in_phrase[DESCRLEN+1];
  int i=0,j=0, k, c, pid, query_err, query_len, in_len, f_flag,look_more=0;
  struct pixrect *pr;
  colormap_t cm;
  char ISfn1[FILENAMELEN+1];
  char descr[DESCRLEN+1];
  int show_pid, wait_pid;
  int  IServror;
  STR_path file_name;
  char ISdescr1[DESCRLEN+1];
  cm.type = RMT_NONE;
  cm.length = 0;
  cm.map[0] = NULL;
  cm.map[1] = NULL;
  cm.map[2] = NULL;
  /* this is absolutely necessary!!!! Otherwise pr_load_colormap might
     not allocate storage for the colormap, if the garbage found in
     the cm structure seems to make sense. The result, of course, is
     segmentation fault. This bug was very hard to find. */
   /*  exec sql select PIXRECT (i_image), COLORMAP (i_image),
                      DESCRIPTION (i_image)
              into :pr, :cm, :descr
                 from image
                 where i_id = :imageno;
     This Image-SQL statement is transformed into the following
     sequence of statements by the preprocessor:
   */
{
        IIsqInit ((char *)0);
      IIwritedb("retrieve unique(c=(count(");
        IIwritedb("result");
        IIwritedb(".");
```

```
        IIwritedb(satt[imageno].a_name);
            IIwritedb("))");
            IIsqRinit((char *)0);
            if (IIerrtest()==0) {
                    if (IInextget() !=0) {
                            IIretdom(1,30,4,&c);
                    }
                    IIsqFlush((char *)0);
            }
}
{
if (IIcsrOpen((char *)0,"cursor_output1","db1",0,media_name) != 0) {
   IIwritedb("retrieve(ISfn1=");
   IIwritedb(media_name);
   IIwritedb(".");
   IIwritedb("f_id,ISdescr1=");
   IIwritedb(media_name);
   IIwritedb(".descrp");
   IIwritedb(")where ");
   IIwritedb(media_name);
   IIwritedb(".i_id=");
   IIwritedb("result.");
   IIwritedb(satt[imageno].a_name);
   IIcsrQuery ((char *)0);
      } /* IIcsropen */
   while (look_more==0) {
          if (IIcsrFetch((char *)0, "cursor_output1","db1") != 0) {
                  IIcsrRet(1,32,0,ISfn1);
                  IIcsrRet(1,32,0,ISdescr1);
          for (i=0;i<MAX_PATH+1;i++) {
                  if (ISfn1[i]==32) {
                          file_name[i]=0;
                  }
                  else {
                          file_name[i]=ISfn1[i];
                  }
              } /* end for */
                  printf("\nRecord no %d filename :%s:",j+1, ISfn1);
                          if ((img_file=fopen(file_name,"r"))==NULL)
                          {
                                  printf("\n%s", file_name);
                                  printf("\nThe file cannot be opened !!\n");
                                  putchar('\007');
                          }
                          else {
                                  pr=pr_load(img_file, &cm);
                                  if (pr==NULL) {
                          printf("\nThe file does not contain proper image");
                                          putchar('\007');
                                          }
                                  else {
                                          printf("\nShow image ....");
                                          present_photo(j+1,pr,&cm,ISdescr1);
                                  } /* end else */
                          } /* end else */
```

```
                fclose(img_file);
        }
        printf("\n");
IIcsrEFetch((char *)0);
        j++;
        if (j==c) {
                look_more = 1;
                };
    }
IIcsrClose((cha  *)0,"cursor_output1","db1");
  }
}
/*************************************************************************/
/* This procedure search through the media relation and get the   */
/* file name that match with the result table and send to the     */
/* play sound procedure                                           */
/*************************************************************************/
display_sound (soundno,tupleno)
int soundno;
int tupleno;
{
 char answer, repeat, yes_no_answer ();
 char query_phrase[DESCRLEN+1],
     in_phrase[DESCRLEN+1];
 int i=0,j=0, k, c, pid, query_err, query_len, in_len, f_flag,look_more=0;
 int show_pid, wait_pid;
 int  ISerror;
 STR_path file_name;
 char ISfn1[FILENAMELEN+1];
 char ISdescr1[DESCRLEN+1];
  {
        IIsqInit ((char *)0);
    IIwritedb("retrieve unique(c=(count(");
        IIwritedb("result");
        IIwritedb(".");
    IIwritedb(satt[soundno].a_name);
        IIwritedb(")))");
        IIsqRinit((char *)0);
        if (IIerrtest()==0) {
                if (IInextget() !=0) {
                        IIretdom(1,30,4,&c);
                }
                IIsqFlush((char *)0);
        }
   }
if (IIcsrOpen((char *)0,"cursor_output1","db1",0,media_name) != 0) {
  IIwritedb("retrieve(ISfn1=");
  IIwritedb(media_name);
  IIwritedb(".");
  IIwritedb("f_id,ISdescr1=");
  IIwritedb(media_name);
  IIwritedb(".descrp");
  IIwritedb(")where ");
  IIwritedb(media_name);
  IIwritedb(".s_id=");
```

```
    IIwritedb("result.");
    IIwritedb(satt[soundno].a_name);
    IIcsrQuery ((char *)0);
       } /* IIcsropen */
    while (look_more==0) {
            if (IIcsrFetch((char *)0, "cursor_output1","db1") != 0) {
                    IIcsrRet(1,32,0,ISfn1);
                    IIcsrRet(1,32,0,ISdescr1);
            for (i=0;i<MAX_PATH+1;i++) {
                    if (ISfn1[i]==32) {
                            file_name[i]=0;
                    }
                    else {
                            file_name[i]=ISfn1[i];
                    }
               }
                    printf("\nRecord no %d ",j+1);
              printf("\nPlay the sound ? (y/n) :: ");
                    if (yes_no_answer() == 'y') {
                play_sound(file_name);
                            }
            printf("\n");
            IIcsrEFetch((char *)0);
            j++;
            if (j==c) {
                    look_more = 1;
            }
         } /* IICSRFECCH */
       } /* end while */
IIcsrClose((char *)0,"cursor_output1","db1");
}  /* end of display_sound () */
/*************************************************************************/
present_photo2 (number, pixels, colormap, description)
int number;
struct pixrect *pixels;
colormap_t *colormap;
char *description;
{
 char answer, yes_no_answer ();
 int i, error, pid;
 Frame frame;
 Canvas canvas;
 Pixwin *pw;
 printf ("Number: %d\n", number);
 printf ("Description:\n>>%s<<\n\n", description);
 answer = 'y';
 if (answer == 'n')
   return (0);
 else {
   pid = fork ();
   if (pid < 0) {
     printf ("Starting display process failed\n\n");
     return (-1);
   }
   if (pid > 0)   /* this is parent process */
```

```
  return (pid);
if (colormap == NULL) {
  printf ("Cannot show it - no colormap.\n\n");
  exit (1);
}
frame = window_create (NULL, FRAME,
                       FRAME_LABEL,        "IMAGE",
                       FRAME_NO_CONFIRM,   TRUE,
                       WIN_WIDTH,          pixels->pr_size.x + 20,
                       WIN_HEIGHT,         pixels->pr_size.y + 50,
                       WIN_ERROR_MSG,      "Cannot create window.",
                       0);
if (frame == NULL) {
  printf ("Cannot create frame\n\n");
  exit (1);
}
canvas = window_create (frame, CANVAS,
                        WIN_WIDTH,         pixels->pr_size.x,
                        WIN_HEIGHT,        pixels->pr_size.y,
                        0);
if (canvas == NULL) {
  printf ("Cannot create canvas\n\n");
  exit (1);
}
pw = canvas_pixwin (canvas);
if (pw == NULL) {
  printf ("pixwin is NULL\n\n");
  exit (1);
}
window_fit (frame);
if (colormap->type == RMT_EQUAL_RGB
      && colormap->length > 0) {
  pw_setcmsname (pw, "photo");
  if (error = pw_putcolormap (pw, 0, colormap->length,
                colormap->map[0], colormap->map[1], colormap->map[2])) {
      printf ("Cannot load colormap.\n");
      printf ("error code = %d\n", error);
      printf ("type = %d\nlength = %d\n", colormap->type, colormap->length);
  /* for (i = 0; i < colormap->length; i++) {
       printf (" %x %x %x\n", *(colormap->map[0] + i),
              *(colormap->map[1] + i), *(colormap->map[2] + i));
      } */
      exit (1);
  }
}
else {
  printf ("Cannot show photo - colormap not appropriate.\n\n");
  exit (1);
}
if (pw_write (pw, 0, 0, pixels->pr_size.x, pixels->pr_size.y,
              PIX_SRC,
              pixels, 0, 0))
  printf ("Cannot write image to screen.\n\n");
else
  window_main_loop (frame);
```

```
  window_destroy (frame);
  exit (0);
 }  /* of (answer = 'y'), showing the photo */
 return (0);
}
/*************************************************************************/
/* This procedure create the embeded psudo extended SQL for user  */
/* display on the screen                                          */
/*************************************************************************/
void processquery2()
{
        char a;
        int i,j,k;
        STR_name media_att;
        int medianum=0;
        int image_select=0; /* For the choose of the extra attribute of type image */
        int snd_select=0; /* For the choose of extra type sound */
  /* For test purpose only */
  for (j=0;j<numcon;j++) {
    printf("\nGroup %d Att %s Atttype %d Con %s",att_group[j],att[j],contype[j],con[j]);
        if (contype[j]==1) {
                printf("\nCREATE TABLE M%d AS SELECT i_id FROM %s WHERE CONTAIN
(%s)", j,att[j],con[j]);
                image_select =1;
                }
        if (contype[j]==2) {
                printf("\nCREATE TABLE M%d AS SELECT s_id FROM %s WHERE CONTAIN
(%s)", j,att[j],con[j]);
                snd_select=1;
        }
  }
  /* End test */
        printf("\nProcess Ingres Interface in the database");
        if (icond==0) {
                printf("\nProcess only formatted data");
        printf("\n\nExec SQL Select ");
        for (i=0;i < n;i++) {
                printf("%s.%s",satt[i].t_name,satt[i].a_name);
                if (i < n-1) {
                        printf(",");
                        }
        } /* End for */
        printf("\nFrom ");
        for (i=0;i < m;i++) {
                printf("%s",stab[i].t_name);
                if (i < m-1) {
                        printf(",");
                        }
        }
        if (cond==1) {
                printf("\nWhere ");
        if (numcon == 1) {
             gcond=0;
             numgroup=0;
        }
```

```
        if (m>1) {
                printf("( %s ) and ", join_condition);
        }
        if (numgroup >= 1) {
                printf("(");
        }
        k=0;
        if (m>1) {
                printf("( ");
        }
        if (gcond==1) {
        for (i=0;i<=numgroup;i++) {
        for (j=group_count[k].begingroup;j < group_count[k].endgroup;j++) {
                if (contype[j]==1) {
                printf("Contain (%s.%s,%s) ",tab[j],att[j],con[j]);
                }
                if (contype[j]==2) {
                printf("Contain (%s.%s,%s) ",tab[j],att[j],con[j]);
                }
                else {
                printf(" %s.%s %s ",tab[j],att[j],con[j]);
                }
                if (j!=group_count[i].endgroup-1) {
                        printf(" and ");
                        }
                }
                k=k+1;
                if (numgroup >= 1) {
                        printf(")");
                        if (k <= numgroup) {
                                printf(" or (");
                                }
                }
           }
        }
        if (numgroup == 0) {
                if (contype[0]==1) {
                printf("Contain (%s.%s,%s) ",tab[0],att[0],con[0]);
                }
                else {
                        printf(" %s.%s %s ",tab[0],att[0],con[0]);
                }
        }
        if (m>1) {
                printf(" )");
        }
} /* End if condition */
        }
        else
        {
        for (i=0;i <= numgroup;i++) {
                printf("\nprocess group %d", i);
        printf("\nExec sql create table G%d as JOIN f%d and m%d ", i,i,i);
                printf("\nCREATE TABLE f%d as SELECT ",i);
        for (i=0;i<n-1;i++) {
```

```
        printf("%s.%s, ",satt[i].t_name,satt[i].a_name);
}
printf("%s.%s ",satt[i].t_name,satt[i].a_name);
        printf("\nFrom ");
        for (j=0;j < m;j++) {
                printf("%s",stab[j].t_name);
        if (j < m-1) {
                printf(",");
                }
        } /* End from */
printf ("\nWhere ");
        if (m>1) {
                printf("( %s ) and ( ", join_condition);
        }
for (j=group_count[i].begingroup;j < group_count[i].endgroup;j++) {
        if (contype[j]==1) {
         printf(" (%s in select i_id from M%d) ",att[j],j);
}
        if (contype[j]==2) {
         printf(" (%s in select s_id from M%d) ",att[j],j);
}
        else {
        printf(" %s %s ",att[j],con[j]);
        }
        if (j!=group_count[i].endgroup-1) {
                printf(" and ");
                }
        }
        k=k+1;
        if (numgroup >= 1) {
                printf(")");
                if (k <= numgroup) {
                        printf(" or (");
                        }
   }
}
if (m>1) {
        printf(" ) ");
        }
}
if (numgroup > 0) {
        printf ("\nEXEC SQL CREATE TABLE OUTPUT AS SELECT ALL FROM ");
        for (i=0;i< numgroup;i++) {
                printf ("G%d or ",i);
                }
        printf("G%d", i);
} /* End if more than one group */
/* Print out the data */
printf("\nSELECT ");
for (i=0;i<n-1;i++) {
printf("%s, ",satt[i].a_name);
}
printf("%s ",satt[i].a_name);
printf("\nFROM OUTPUT");
gcond = 0;
```

```
}
/*************************************************************************/
/* This procedure create the embeded psudo extended SQL for user  */
/* display on the screen                                          */
/*************************************************************************/
void processquery()
{
        char a;
        int i,j,k;
        int medianum=0:
        number_media=0;
        printf("\n\nSelect ");
        for (i=0;i < n;i++) {
                printf("%s.%s",satt[i].t_name,satt[i].a_name);
                if (i < n-1) {
                        printf(",");
                        }
                }
        printf("\nFrom ");
        for (i=0;i < m;i++) {
                printf("%s",stab[i].t_name);
                if (i < m-1) {
                        printf(",");
                        }
                }
        if (cond==1) {
                printf("\nWhere ");
        if (numcon == 1) {
             gcond=0;
             numgroup=0;
        }
        if (numgroup >= 1) {
                printf("(");
        }
        k=0;
        if (gcond==1) {
                for (i=0;i<=numgroup;i++) {
                for (j=group_count[k].begingroup;j < group_count[k].endgroup;j++) {
                        if ((contype[j]==1)||(contype[j]==2)) {
                                printf("Contain (%s,%s) ",att[j],con[j]);
                                strcpy(media_att[number_media],att[j]);
                                number_media=number_media+1;
                        }
                        else {
                                printf(" %s %s ",att[j],con[j]);
                        }
                        if (j!=group_count[i].endgroup-1) {
                                printf(" and ");
                        }
                } /* END FOR J */
                k=k+1;
                if (numgroup >= 1) {
                        printf(")");
                        if (k <= numgroup) {
                                printf(" or (");
```

```
                                        }
                        }
                        } /* End second for */
                }
                /* only one condition process */
                if (numgroup == 0) {
                        if ((contype[0]==1)||(contype[0]==2)) {
                        printf("Contain (%s,%s) ",att[0],con[0]);
                        strcpy(media_att[number_media],att[0]);
                        number_media=number_media+1;
                        }
                        else {
                                printf(" %s %s ",att[0],con[0]);
                        }
                }
   } /* End if condition */
        processquery2();
}
/*************************************************************************/
/* This procedure get the query description for the media attribute*/
/* from the user phrase by phrase                                        */
/*************************************************************************/
char process_icon()
{
 char answer, repeat, yes_no_answer ();
 char query_phrase[DESCRLEN+1],
     in_phrase[DESCRLEN+1];
 int i, query_err, query_len, in_len, f_flag;
 char descr[DESCRLEN+1];
 int show_pid, wait_pid;
 int imageno;
 icond = 1;
 do
 {
        query_err = 0;
        query_len = 0;
        query_phrase[0] = '\0';
        printf("\nPlease enter your query description (one phrase per line;\n\
end with empty line):\n");
        do /* until query_phrase input */
        {
          i = 0,
          while ( (in_phrase[i++] = getchar()) != '\n' && i < 127 );
          if ( in_phrase[i-1] != '\n' )
          {
            in_phrase[i-1] = '\n';
            printf ("The phrase is too long, it will be shortened\n");
            while ( getchar () != '\n' );
          } /* End if */
          in_phrase[i] = '\0';
          if ( ( in_len = i ) > 1 )
          {
            if ( query_len + in_len < DESCRLEN )
            {
                strcat(query_phrase,in_phrase);
```

```
                query_len = query_len + in_len;
            } /* End if */
            else
            {
                printf("The last phrase extended beyond the maximum \
description length,\nit will be ignored\n");
                break;
            }  /* End else */
          } /* End if */
          if ( !query_len )
            printf("\nAn empty string is not allowed as a query description.\n\
Please type at least a single word:\n");
        } /* End do */
        while ( ( in_len > 1 ) || !query_len ); /* end query_phrase input */
        printf("The query description now is:\n>>%s<<\n\n",query_phrase);  /* print the dscription */
        } while (query_err);
  strcpy(con[numcon],query_phrase);                    /* copy description into condition array */
  process_icon2 (query_phrase,numcon);
}
/*********************************************************************/
/* This procedure accumulate the condition from the user and form  */
/* the group condition of and and or                                 */
/* Mean condition that compose of disjunctive normal form          */
/*********************************************************************/
void gcondition()
{
        int endgroup,i,more,found=FALSE;
        char ans;
        gcond=1;
        endgroup = 0;
        more = 0;
        numcon=0;
        numgroup=0;
        group_count[0].begingroup = numcon;
        while (more != 1)  {
        while (endgroup != 1) {
        for (i=0;i < att_index;i++)
        {
                if (m > 1 ) {                          /* if more than 2 tables in the selection */
                printf("\nEnter table name ");
                gets(tab[numcon]);
                strcpy (table_array[table_index].table_name, tab[numcon]);
                }
                if (m==1) {                            /* if only 1 table just copy the table */
                  strcpy (tab[numcon], stab[0].t_name);
                }
                printf("\nEnter attribute ");        /* attribute for condition */
                gets(att[numcon]);
     att_group[numcon]=numgroup;
                getatttype(tab[numcon], att[numcon],atttype[numcon]);
    if (strcmp(atttype[numcon],"image")==0)       /* check for image condition */
        {
          contype[numcon]=1;
          process_icon();
        }
```

```
    else if (strcmp(atttype[numcon],"sound")==0) /* check for sound condition */
        {
          contype[numcon]=2;
          process_icon();
        }
        else {                                          /* if not media condition then it is formatted data */
                printf("Enter the condition \n");
                gets(con[numcon]);
                contype[numcon]=0;
                printf("\nWhere %s %s",att[numcon],con[numcon]);
           }
        numcon=numcon+1;
        printf("\nEnd group ?");
        ans=yes_no_answer();
        if ((ans==121)||(ans==89)) {
           endgroup=1;
           printf("\nGroup      %d",numgroup);       /* print for checking group */
           printf("\nCondition  %d",numcon);
           i=600;
           }
        } /* End for */
        } /* END WHILE */
        printf("\nEnd condition ?");
        ans=yes_no_answer();
        if ((ans==121)||(ans==89))
        {
           group_count[numgroup].endgroup = numcon;
           endgroup=1;
           more = 1;
           i=0;
        }
        else {
                more =0;
                endgroup=0;
                more = 0;
                i=0;
           group_count[numgroup].endgroup = numcon;       /* assign endgroup and begin */
           numgroup=numgroup+1;
           group_count[numgroup].begingroup = numcon;
           }
    } /* End more */
}
/*************************************************************************/
/* process the array of the variable and generate the query of the SQL*/
/* to process in procedure join                                  */
/*************************************************************************/
void processcondition()
{
        char   ans2,a;
        int i,j;
        cond=1;
        gcond=0;
        printf("\nGroup condition ? (y/n)  ");
        ans2=yes_no_answer();
        if ((ans2==121)||(ans2==89))
```

```
        {
         gcond=1;
         gcondition();
        }
        else
        {
                gcond=0;
                if (m > 1 ) {
                printf("\nEnter table name ");        /* enter table name for condition */
                gets(tab[0]);
                }
                if (m==1) {
                  strcpy (tab[0], stab[0].t_name);
                }
                printf("\nEnter attribute name "); /* enter attribute name for condition */
                gets(att[0]);
                printf("\n%s %s %s", tab[0], att[0], atttype[0]);
                getatttype(tab[0],att[0],atttype[0]);
    if (strcmp(atttype[0],"image")==0)                  /* check for image */
        {
                contype[0]=1;
                process_icon();
        }
     else if (strcmp(atttype[0],"sound")==0)            /* check for sound */
        {
                contype[0]=2;
                process_icon();
        }
        else {
                printf("Enter the condition \n");       /* formatted condition */
                gets(con[0]);
                contype[0]=0;
                printf("\nWhere %s.%s %s",tab[0],att[0],con[0]);
           }
        }
}
/***************      ***************************************************/
/* This procedure print the attribute name of the table assign to     */
/*********************************************************************/
void p_att(tab_name)
STR_name tab_name;
{
        int i,j;
        for (i=0;i<= table_count;i++) {                 /* loop until no morte table */
           if (strcmp(table_array[i].table_name,tab_name)==0) {
                x = i;
                y = table_array[i].att_entry;
                printf("\n%s",table_array[i].table_name); /* print table name */
                        while (y != -1) {
printf("\nAttribute %s data type is %s",att_array[y].att_name,att_array[y].data_type);
                         y = att_array[y].next_index;
                        } /* End while y!=-1 */
                        if (y==-1) {
                                i=500;
                        } /* Exit loop */
```

```
                }  /* End if */
        }  /* End for */
}
/*****************************************************************************/
/* Print out all the tables information on screen                            */
/*****************************************************************************/
void p_table()
{
  int i = 0;
  printf("\t**Table Name**\n");
  for (i = 0; i < table_count; i++)
   {
    printf("\t  %s\n",table_array[table_list[i]].table_name);
    if ((i % 15) == 14)
        {
         printf("\n*RETURN TO CONTINUE*\n");
         while ((c = getchar()) != '\n')
         ;
         printf("\t**Table Name**\n");
        }
   }  /* End of for loop */
}  /* End of print_all_table() */
/*************************************************/
/* Generate the result table for retrieval process */
/* This procedure process the query and condition  */
/* By using the select_array and condition_array   */
/* also group_array                                */
/*************************************************/
void ql_retrieve()
{
    int i,j,k;
   i=0; /* set up index to 0 */
        /* Below is the embeded C code for the SQL C for INGRES */
        /* This is equivalent to the SQL query */
        /*  exec sql select (var1, var2, ...)
                from (table1, table2,...)
                where (condition1 and/or condition2 and/or ...);
        */
   IIsqInit((char *)0);
   IIwritedb("retrieve into result(");
   for (i=0;i<n-1;i++) {
   IIwritedb(satt[i].t_name);
   IIwritedb(".");
   IIwritedb(satt[i].a_name);
   IIwritedb(",");
        } /* end for */
   IIwritedb(satt[i].t_name);
   IIwritedb(".");
   IIwritedb(satt[i].a_name);
   IIwritedb(")");
   if (cond==0) {
        if (m>1) {
                IIwritedb("where ");
                IIwritedb(join_condition);
        }
```

```
}
if (cond==1) {
IIwritedb("where (");
        if (m>1) {
                IIwritedb("(");
                IIwritedb(join_condition);
                IIwritedb(")");
                IIwritedb("and ");
        }
        if (gcond == 1) {
                IIwritedb("(");
        for (i=0;i<=numgroup;i++) {
        for (j=group_count[k].begingroup;j<group_count[k].endgroup;j++) {
         if (contype[j]==0) {
                IIwritedb(tab[j]);
                IIwritedb(".");
                IIwritedb(att[j]);
                IIwritedb(con[j]);
                } /* end if */
         /* for the media condition the query will get the value in the */
         /* intermediate table generate in the procedure process_icond  */
         if (contype[j]==1) {           /* image condition */
                IIwritedb(tab[j]);
                IIwritedb(".");
                IIwritedb(att[j]);
                IIwritedb("=");
                IIwritedb("m");
                IIwritedb(".");
                IIwritedb("i_id");
                }
         /* for the media condition the query will get the value in the */
         /* intermediate table generate in the procedure process_icond  */
         if (contype[j]==2) {           /* sound condition */
                IIwritedb(tab[j]):
                IIwritedb(".");
                IIwritedb(att[j]);
                IIwritedb("=");
                IIwritedb("m");  /* This is the media table followed by the */
                IIwritedb(".");  /* condition number in the query          */
                IIwritedb("s_id");
                }
          /* Between group has the boolean and to be the conjunction */
          if (j != group_count[i].endgroup-1) {
                IIwritedb(" and ");
                }
        }
        k=k+1;
        /* This is suppose for the or boolean but still has some bug */
        if (numgroup >= 1) {
                if (k <= numgroup) {
                IIwritedb (") or (");
                }
           }
        }
                if(gcond==1) {
```

```
                    IIwritedb(")");
                }
            }
            if (numgroup==0) {
             if (contype[0]==0) {
                    IIwritedb(tab[0]);
                    IIwritedb(".");
                    IIwritedb(att[0]);
                    IIwritedb(con[0]);
                    } /* end if */
                    if (contype[0]==1) {
                    IIwritedb(tab[0]);
                    IIwritedb(".");
                    IIwritedb(att[0]);
                    IIwritedb("=");
                    IIwritedb("m");
                    IIwritedb(".");
                    IIwritedb("i_id");
                        }
            if (contype[0]==2) {
                    IIwritedb(tab[0]);
                    IIwritedb(".");
                    IIwritedb(att[0]);
                    IIwritedb("=");
                    IIwritedb("m");
                    IIwritedb(".");
                    IIwritedb("s_id");
                    }
            } /* end if no group */
            IIwritedb(")");
            } /* end if condition */
IIsqSync(0,(char *)0);/* send the signal to INGRES to execute the function */
}
/*************************************************************************/
/* This procedure set the cursor point to result table and print  */
/* After finish the formatted data then go to the media data      */
/* The media data begin with image and then sound                 */
/*************************************************************************/
void ql_printdata()
{
 int c=0,j=0,k=0,l=0,temp;
 char char_value[21],a;
 char file_name[20];
 int  integer_value,media_value,found,media1_value;
 float real_value;
  int i=0,select=0;
/* # line 3169 "db.sc" */   /* select */
  {
   IIsqInit((char *)0);
   IIwritedb("retrieve(c=(count(");
   IIwritedb("result");
   IIwritedb(".");
   IIwritedb(satt[0].a_name);
   IIwritedb("))))");
   IIsqRinit((char *)0);
```

```
  if (IIerrtest() == 0) {
    if (IInextget() != 0) {
      IIretdom(1,30,4,&c);
    } /* IInextget */
    IIsqFlush((char *)0);
  } /* IIerrtest */
 }
l=0;
printf("\nThere are %d records that match the query",c);
/* # line 3171 "db.sc" */  /* host code */
  if (IIcsrOpen((char *)0,"cursor_output","db1",0,"result") != 0) {
    IIwritedb("retrieve(");
    for (select=0;select<n-1;select++) {
    IIwritedb(satt[select].a_name);
    IIwritedb("=");
    IIwritedb("result.");
    IIwritedb(satt[select].a_name);
    IIwritedb(",");
          }
    IIwritedb(satt[select].a_name);
    IIwritedb("=");
    IIwritedb("result.");
    IIwritedb(satt[select].a_name);
    IIwritedb(")");
    IIcsrQuery((char *)0);
  } /* IIcsrOpen */
printf("\n");
look_more=0;
l=0;
if (c==0) {
          look_more=1;
          }
/* Fetch the cursor to the result relation which is the intermediate table
  hold the result from the query, then print out the tuple one at a time
  until no more record to print to the user */
while (look_more == 0) {
  if (IIcsrFetch((char *)0,"cursor_output","db1") != 0) {
                printf("record id %d \t",l+1);
      for (i=0;i<n;i++) {
          if (strcmp(satt[i].data_type,"c20")==0) {
        IIcsrRet(1,32,0,char_value);
                printf("%s :  %s",satt[i].a_name,char_value);
                }
          if (strcmp(satt[i].data_type,"integer")==0) {
        IIcsrRet(1,30,4,&integer_value);
                printf("%s :  %d  ",satt[i].a_name,integer_value);
                }
          if (strcmp(satt[i].data_type,"float")==0) {
        IIcsrRet(1,31,4,&real_value);
                printf("%s :  %8.2f  ",satt[i].a_name,real_value);
                }
          if (strcmp(satt[i].data_type,"image")==0) {
        IIcsrRet(1,30,4,&media_value);
                printf("%s id is %d  ",satt[i].a_name,media_value);
                }
```

```
            if (strcmp(satt[i].data_type,"sound")==0) {
          IIcsrRet(1,30,4,&media1_value);
                  printf("%s %d",satt[i].a_name,media1_value);
                  }
          } /* end for select < n*/
          printf("\n");
      IIcsrEFetch((char *)0);  /* fetch the next record to the cursor */
          l++;  /* increment l as the counter */
          if (l==c) {   /* check if no more data to print */
                  look_more =1;  /* exit of the loop */
                  }
    } /* IIcsrFetch */
} /* end while */
IIcsrClose((char *)0,"cursor_output","db1"); /* close the cursor */
          printf("Press any key to continue ..");
          /* stop before change to the next function so
            the user can see the result on screen, until he hit any key */
          a= getchar();
/* this for the check for the media selection */
if (c==0) {
          i=9999; /* if no record for the media data not process any thing */
          }
       for (i=0;i<n;i++) {
            if (strcmp(satt[i].data_type,"image")==0) {
              strcpy(table_array[table_index].table_name, satt[i].t_name);
                  found = check_table_name(); /* search for the media name */
                  table_cursor = table_entry;
                  strcpy(media_name,satt[i].a_name);
                  get_media_name();
                  display_photo(i,j);
                  /* display photo search for the image relation
                    that match the result tuple then open the file */
                  }
            if (strcmp(satt[i].data_type,"sound")==0) {
                  printf("\nSound management");
              strcpy(table_array[table_index].table_name, satt[i].t_name);
                  found = check_table_name();
                  table_cursor = table_entry;
                  strcpy(media_name,satt[i].a_name);
                  get_media_name();
                  display_sound(i,j);
                  /* play sound search for the sound relation
                    that match the result tuple then open the file */
                  }
          } /* end for select < n*/
          printf("\n");
/* Drop table result after finished print  */
  {
   IIsqInit((char *)0);
   IIwritedb("destroy result");
   IIsqSync(0,(char *)0);
  }
}
/*****************************************************************************/
/* The main procedure for the retrieve operation                             */
```

```
/* m and n is the parameter for table and attribute repectively        */
/* For retrieve table name and attribute name from the user            */
/*************************************************************************/
void retrieve()
{
        int i,j,x,y,z,found=0;
        char table_name[20],attname[20],att_type[20],Ans,More,a;
        init();
        /* Select table */
        for (i=0;i<100;i++) {
                buff[i] = '\0';/* assign null value or end of string to buffer*/
        }
        m=0;
        i=0;
        k=0;
        strcpy(buff,"?");
        while (strcmp(buff,"?")==0) { /* select loop for help function */
        printf("\nSelect the table(s) saparate by comma <,> : ");
        gets(buff);
        if (strcmp(buff,"?")==0) {
                p_table();
                }
        if (strcmp(buff," ")==0) {
                return;
                }
                i=0;
        } /* end while buff == 0 */
        while (i<=table_count) {    /* check loop with the maximum number table */
          for (j=0;j<13;j++)                /* each table has less than or equal to 12 char only */
          {
                if (buff[k]==44) {
                        j=55;
                        k=k+1;
                        i=i+1;
                }
                else {
                        if (buff[k]==0) {  /* if null value in buffer (end of string) */
                                m=i+1;
                                j=55;
                                i=1000;
                        }
                        stab[i].t_name[j]=buff[k];
                        k=k+1;
                }
          }
        } /* End while */
        for (i=0;i<m;i++) {
            strcpy(table_array[table_index].table_name, satt[i].t_name);
                found = check_table_name(); /* search for the media name */
                if (!(found)) {
/* check for the valid table name if not found then return to calling program */
                        putchar('\007');
                        printf("\nTable %s not found please redo again !!!" ,satt[i].t_name);
                        printf("\nPress any key to continue !!");
                        a=getchar();
```

```
                return;
                } /* end else */
} /* end for loop */
/* Specify the join condition if there are more than 2 table select */
if (m > 1) {
strcpy(join_condition,"?");
while (strcmp(join_condition,"?")==0) {
        printf("\nPlease enter your join condition : ");
        gets(join_condition);
        if (strcmp(join_condition,"?")==0) {
                for (i=0;i<m;i++) {
                        printf("\nTable %s ", stab[i].t_name);
                        p_att(stab[i].t_name);
                } /* end for loop */
        if (strcmp(join_condition," ")==0) {
                return;
              }
        } /* end if need help for join */
 } /* end while */
} /* end if more than 1 table select */
/* Select attribute */
for (i=0;i<100;i++) {
        buff[i] = '\0';
}
i = 0;
j = 0;
k = 0;
x = 0;
z = 0;
/* Select attribute for one table at a time */
for (y=0;y<m;y++) {
   printf("\nTable %s ", stab[y].t_name);
   strcpy(buff,"?");
   while (strcmp(buff,"?")==0) {
       printf("\nSelect the attribute(s) separate by comma <,> : ");
        gets(buff);
       if (strcmp(buff,"?")==0) {
            p_att(stab[y].t_name);
         } /* end if buff == "?"  */
        if (strcmp(buff," ")==0) {
        /* exit if user put space only to the buffer */
                return;
              }
    } /* end while need help */
   while (i < 100) {
    for (j=0;j<13;j++)
    {
        if (buff[k]==44) {
                j=55;
                k=k+1;
                i=i+1;
                x=x+1;
        }
        else {
                if (buff[k]==0) {
```

```
            strcpy(satt[x].t_name, stab[y].t_name);
                                    n=x+1;
                                    j=55;
                                    i=1000;
                            printf("%d",n);
                            }
                            satt[x].a_name[j]=buff[k];
                            k=k+1;
                    } /* end else */
              strcpy(satt[x].t_name, stab[y].t_name);
              } /* end for j < 13 */
            }
            x=x+1;
            k=0;
            for (i=0;i<100;i++) {
                    buff[i] = '\0';
            }
            i=0;
            } /* End select attribute for each table go to the next table */
            for (i=0;i<n;i++) {
                    printf("\n%s.%s", satt[i].t_name,satt[i].a_name);
                    getatttype(satt[i].t_name,satt[i].a_name,satt[i].data_type);
            }
            printf("\n");
            cond=0;
            printf("\nAny condition ? (y/n) ");
            Ans=yes_no_answer();
            if ((Ans==121)||(Ans==89))
            {
             cond=1;
             processcondition();
            }
            processquery();
            ql_retrieve();
            ql_printdata();
} /* End procedure */
/*****************************************************************************/
/* Main program for MDBMS                                                    */
/*****************************************************************************/
main()
{
  int wrong_descrp = TRUE;
  int i=0,j=0;
  char Ans, a;
  char function = 0;
  char choice = '?';
  printf("\nConnect to database ");
   printf("\nwait ..... ");
  {
        IIsqConnect (&sqlca,0,"virgo::mdb", (char *)0); /* this code use for connect to the database */
  }
  if (sqlca.sqlcode != 0)     /* error in connection to database */
  {
        printf("\nSorry, but we cannot connect to the database at this time!\n\
It could be that you are execute the program in the wrong system.\n\
```

```
Please write down your code and give them to the administrator:\n\
sqlca.sqlcode = %ld\n", sqlca.sqlcode);
        exit(1);
        }
  load_data();   /* load catalog from the file into memory */
/* # line 3504 "db.sc" */  /* destroy */
  {                                 /* Drop table result in database */
   IIsqInit((char *)0);
   IIwritedb("destroy result");
   IIsqSync(0,(char *)0);
  }
  clr_scr();
  while (choice != '0')
  {
    choice = user_choice();         /* print the choice for user select on screen */
    switch(choice)                  /* User select case */
    {
        case '1' :                  /* create table */
                clr_scr();
                create_table();
                display_info();
                ql_create_table();
                store_data();       /* save data back in the file */
                break;
        case '2' :                  /* insert tuple */
                clr_scr();
                insert_tuple();
           wrong_descrp = TRUE;
           while (wrong_descrp)
             {
              display_tuple();
             wrong_descrp = check_media_descrp();
             }
           if (!wrong_descrp)
             {
                    printf("\n\nHit RETURN to Continue!!");
            while ((c = getchar()) != '\n')
              ;
             };
           store_data();
           ql_insert_tuple();
                break:
        case '3' :                  /* retrieve */
                clr_scr();
           retrieve();
                break;
        case '4' :                  /* deletion */
                clr_scr();
                printf("Your selection %c is: ", choice);
                printf("Delete \n");
                while ((c = getchar()) != '\n')
                ; /* Not return do nothing */
                break;
        case '5' :                  /* update or modify */
                clr_scr();
```

```
                printf("Your selection %c is: ", choice);
                printf("Modify \n");
                while ((c = getchar()) != '\n')
                ; /* Not return do nothing */
                break;
        case '6' :                      /* Test purpose now */  /****/
                clr_scr();                                      /****/
                print_out_data();                                 /****/
                break;                                          /****/
        case '0' :
                clr_scr();
                printf("Thank you for using MDBMS \n");
                while ((c = getchar()) != '\n')
                ; /* Not return do nothing */
                break;
    } /* End of switch */
  } /* End of while choice != '0' */
/* # line 1895 "dbpei.sc" */          /* disconnect */
  {
   IIsqExit(&sqlca);
  }
/* # line 1896 "dbpei.sc" */          /* host code */
} /* End of main() */
```

REFERENCES

[AT90] Atila, Y.V., " Design and Implementation of a Multimedia DBMS: Sound Management Integration", M.S. Thesis, Naval Postgraduate School, Department of Computer Science, Monterey, CA, in preparation.

[Be85] Bertino, E., Gibbs, S., Rabitti, F., Thanos, C., and Tsichritzis, D., "Architecture of a Multimedia Document Server," in Proc. 2nd ESPRIT technical Week (Brussels, Sept. 1985), 1985.

[Be86] Bertino, E., Gibbs, S., Rabitti, F., Thanos, C., and Tsichritzis, D., "A Multimedia Document Server," in Proc. 6nd Japanese Advanced Database Symposium (Tokyo, Aug. 1986), Information Processing Society of Japan, 1986, pp. 123-134.

[BRG88] Bertino, E., Rabitti, F., Gibbs, S., "Query processing in a Multimedia Document System," ACM Trans. on Office Information Systems, vol. 6, no. 1, Jan. 1988, pp. 1-41.

[Ch86] Chrisodoulakis. S., Theodoridou, M., Ho, F., Papa, M., and Pathria, A., "Multimedia Document Presentation, Information Extraction, and Document Formation in MINOS: A Model and a System," ACM Trans. on Office Information Systems, vol. 4, no. 4, Oct. 1986, pp. 345-383.

[Du90] Dulle, J., "The Scope of Descriptive Captions for Use in a Multimedia Database System", M.S. Thesis, Naval Postgraduate School, Department of Computer Science, Monterey, CA, June 1990.

[KKS87] Kosaka, K., Kajitani, K., and Satoh, M., "An Experimental Mixed-Object Database System," in Proc IEEE CS Office Automation Symposium (Gaithersburg, MD, April 1987), IEEE CS Press, order no. 770, Washington 1987, pp. 57-66.

[LM88] Lum, V.Y., and Meyer-Wegener, K., "A Conceptual Design for a Multimedia DBMS for Advanced Applications," report no. NPS52-88-025, Naval Post Graduate School, Monterey, CA, August 1988.

[LM90] Lum, V.Y. and Meyer-Wegener, K., "An Architecture for a Multimedia Database Management System Supporting Content Search," Advances in Computing and Information, Proceedings of the Intemational Conference on Computing and Information (ICCI'90), Niagra Falls, Canada, May 23-26, 1990 and to appear in Lecture Notes in Computer Science, Springer Verlag.

[MLW89] Meyer-Wegener, K., Lum, V.Y., and Wu, C.T., "Image Database Management in a Multimedia System," in Visual Database Systems, (IFIP TC2/G2.6 Working Conference, Tokyo, Japan, April 3-7, 1989), Ed. T.L. Kun :, North-Holland, Amsterdam 1989, pp. 497-523.

[PE90] Pei, S., "Design and Implementation of a Multimedia DBMS: Catalog Management, Table Creation and Data Insertion", M.S. Thesis, Naval Postgraduate School, Department of Computer Science, Monterey, CA, in preparation.

[Sa88] Sawyer, G., "Managing Sound in a Relational Multimedia database System," M.S. Thesis, Naval Postgraduate School, Department of Computer Science, Monterey, CA, December 1988.

[SR86] Stonebraker, M. and Rowe L., "The Design of POSTGRES", Proc. SIGMOD Conference, Washington D.C., May 1986.

[Th88] Thomas, C.A., "A Program Interface Prototype for a Multimedia Database Incorporating Images", M.S. Thesis, Naval Postgraduate School, Department of Computer Science, Monterey, CA, December 1988.

[WNTA89] Wu, C.T., Nardi P., Turner, H., Antonopoulos D., "Argos next generation shipboard information management system", Report no. NPS52-90-006, Naval Postgraduate School, Department of Computer Science, Monterey, CA, December. 1989.

[WK87] Woelk, D. and Kim, W. "Multimedia Management in an Object-Oriented Database System", Proc. 13th Int. Conf on VLDB, Brighton (England), Sept. 1987.

INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Technical Information Center Cameron Station Alexandria, Virginia 22314-6145	2
2. Library, Code 52 Naval Postgraduate School Monterey, California 93943-5100	2
3. Center for Naval Analyses 4401 Ford Ave. Alexandria, Virginia 22302-0268	1
4. John Maynard Code 042 Command and Control Departments Naval Ocean Systems Center San Diego, CA 92152	1
5. Dr. Sherman Gee ONT-221 Cheif of Naval Research 880 N. Quincy Street Arlington, VA 22217-5000	1
6. Leah Wong Code 443 Command and Control Departments Naval Ocean Systems Center San Diego, CA 92152	1