NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

DESIGN AND IMPLEMENTATION OF A MULTIMEDIA DBMS : COMPLEX QUERY PROCESSING

by

Huseyin Aygun

September 1991

Thesis Advisor: Vincent Y. Lum

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: Computer Science Dept. Naval Postgraduate School; 6b. OFFICE SYMBOL (if applicable): CS37	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 NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
DESIGN AND IMPLEMENTATION OF A MULTIMEDIA DBMS: COMPLEX QUERY PROCESSING (U)

12. PERSONAL AUTHOR(S)
Huseyin Aygun

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Master's Thesis	FROM 08/89 TO 09/91	September 1991	240

16. SUPPLEMENTARY NOTA 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 United States Government.

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Multimedia Database Management System, Multimedia, DBMS, MDBMS, Complex Query, Nexted Query, Multiple Selection, Aggregate Function.

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

Traditional Database Management Systems (DBMS) are capable of managing only alphanumeric data. The Multimedia Database Management (MDBMS) Prototype started at the Computer Science Department of Naval Postgraduate School in 1988 made it possible to capture, store, manage, retrieve and present different media information such as image and sound by using the current, modern computer technology. In the existing MDBMS, if a query references only formatted data, it is passed to INGRES directly, but if a query includes media data, then the query is 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. This thesis will concentrate on complex queries involving nesting conditions and multiple selections which are not supported by the existing MDBMS prototype.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: [X] UNCLASSIFIED/UNLIMITED [ ] SAME AS RPT. [ ] DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION: UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL: Vincent Y. Lum	22b. TELEPHONE (Include Area Code): (408) 646-2175; 22c. OFFICE SYMBOL: CsLm

Approved for public release; distribution is unlimited

DESIGN AND IMPLEMENTATION OF A MULTIMEDIA DBMS: COMPLEX QUERY PROCESSING

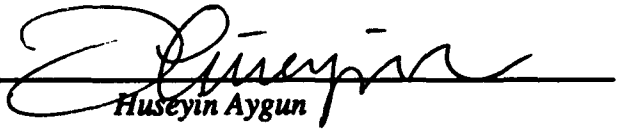
by
Huseyin Aygun
Lieutenant JG, Turkish Navy
B.S., Turkish Naval Academy, 1985

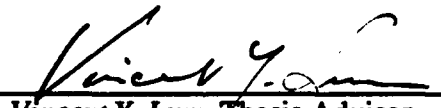
Submitted in partial fulfillment of the
requirements for the degree of

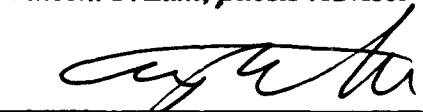
MASTER OF SCIENCE IN COMPUTER SCIENCE

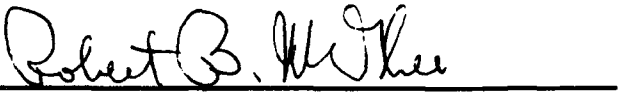
from the

NAVAL POSTGRADUATE SCHOOL
September 1991

Author: 
Huseyin Aygun

Approved By: 
Vincent Y. Lum, Thesis Advisor


C. Thomas Wu, Second Reader


Robert B. McGhee, Chairman,
Department of Computer Science

ABSTRACT

Traditional Database Management Systems (DBMS) are capable of managing only alphanumeric data. The Multimedia Database Management System (MDBMS) prototype started at the Computer Science Department of Naval Postgraduate School in 1988 made it possible to capture, store, manage, retrieve and present different media information such as image and sound by using the current, modern computer technology. In the existing MDBMS, if a query references only formatted data, it is passed to Ingres directly, but, if a query includes media data, then the query is 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. This thesis will concentrate on complex queries involving nesting conditions and multiple selections which are not supported by the existing MDBMS prototype.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

I am grateful to Dr. Klaus Meyer-Wegener, Dr. C. Thomas Wu, Gregory R. Sawyer and Cathy A. Thomas, Su-Cheng Pei, Wuttipong Pongswuan and Yavuz Atila who have made great contributions to the development of the MDBMS system and have thus provided me the opportunity to work on this important database project. I would also like to thank my thesis companions Rosemary Stewart and Chuck Peabody, with whom I worked together as a team to design the current MDBMS prototype.

I would like to express my sincere thanks to Dr. Vincent Y. Lum and Daniel Keim for all their support and encouragement in the conception and preparation of this thesis.

I. INTRODUCTION

A. BACKGROUND

Multimedia database management systems (MDBMS) manage multimedia data such as image data and sound data in addition to formatted data. Multimedia database management systems are currently attracting a lot of attention because of the demands of the new applications and the advances of the technology, making it possible to capture and store multimedia data in computers. Multimedia data broadens the communication between the computer system and the user. Many applications, military, publishing, or instructional, routinely need multimedia data. Although the cost of the hardware required to handle multimedia data is decreasing rapidly, the software needed to manage such multimedia data is lacking or does not match the needs.

Studies on multimedia database management systems started in Computer Science Department of Naval Postgraduate School in 1988. Besides storing, managing and retrieving different media information, the MDBMS prototype also manages the interrelationships between formatted and media data. Text, graphics, images, sound, signals and video are the elements of multimedia data. What is common about them is that they all require rather large storage space and consist of a large and varying number of small items, like characters, pixels, lines or frequency indicators stored together in some way to form a unit. They all have a more complicated structure than formatted data, and require the use of Abstract Data Type (ADT) concept. With this approach, image,

sound, signal, text and graphic data will be treated as new data types. Any attribute of an object can have one of these types. Currently can use the database operations, create, retrieve, modify and delete, on these new media data "values".

In our current prototype system we use separate files to store the media objects because of their high storage requirement. A media object is the value of a media attribute. An image, for example, is a media object, but it is also the value of the attribute picture, just like "John" being the value of the attribute name in an Employee relation.

The main task of MDBMS is storage and retrieval, but not processing of data. The storage and retrieval of multimedia data should be done by the content of the data, but handling content search is a difficult problem, since it is not possible to use the methods currently done on formatted data structures.

Since automatic recognition of the contents of media data by the computer is not possible using today's technology, the decision was made in the MDBMS project to use natural language descriptions to specify the contents of media data. A Prolog parser was constructed to understand the meaning of the natural language captions describing the content of the media data. When the user makes a query related to multimedia data, the PARSER recognizes syntax and semantics of the natural language description and interacts with the MDBMS to locate the appropriate data items [REF6].

INGRES is used to store and manage the data. However, many of the tables are transparent to the users. For example when the user wants to create a table which includes media data using a SQL statement, the system creates a separate table for each media attribute in addition to the tables for formatted data.

B. RELATED WORK

Besides the MDBMS Project at Naval Postgraduate School, there are a number of researches going on in multimedia data processing around the world. Among those, the MINOS system [REF4] developed by a team at the University of Toronto manages highly structured multimedia objects that consists of attributes as well as the text, image and voice part. Sophisticated browsing and user interface features allow the browsing of the schema as well as synchronized updates. The MCC Database program [REF15, REF16] also undertook several multimedia projects by establishing the database requirements of multimedia applications. They identified requirements for a data model and for the sharing and manipulation of multimedia data. An O-O database management system named ORION has been developed at MCC in Austin/Texas, which contains a Multimedia Information Manager (MIM) for processing multimedia data [REF14]. The IBM Tokyo Research Laboratory has developed two "mixed-object database systems", which are named as MODES1 and MODES2 [REF5]. In Europe there is an ESPRIT project designing a multimedia filing system called MULTOS [REF2, REF3].

Recently multimedia management in the personal computers becomes available by using hypertext and hypermedia. The concept of hypertext is very old; it has been transferred to computer systems since 1960's. Originally intended to manage arbitrarily linked text segments, it has been extended to manage images and sound, and has become "Hypermedia" [REF7]. The hypertext and hypermedia data management in the Macintosh computer with a hypercard application has many users, including the ARGOS project being developed at Naval Postgraduate School [REF13]. The hypertext and hypermedia

data management uses the hierarchical data structure approach, in which users cannot query the data as done in the conventional DBMSs, but have to follow the hierarchical tree structure to process a media. As a result, the users might easily get lost during a process. Additionally, hypertext requires an interpreter to process the user commands. Furthermore, the hypertext and hypermedia data cannot be accessed by other users, as in the database systems, because they are designed to work only on personal computers in the single user environment. MDBMS, which is a DBMS introduced in [REF6] with the extended capability to process the multimedia data, was designed to overcome the restrictions and disadvantages of hypertext and hypermedia systems.

C. THE SCOPE OF THE THESIS

The overall design of the MDBMS prototype was a team effort and is given in the thesis by Wuttipong Pongsuwan [REF10], Yavuz Atila [REF1] and Su-Cheng Pei [REF8] but different parts appear on different levels of details. In [REF10] the retrieval process is given, in [REF1] the management of sound data is described, and in [REF8] table creation and data insertion is given. Modify, delete, graphical user interface design is given in the accompanying thesis [REF12] and [REF9].

An important aspect of an MDBMS is the retrieval process. In the existing MDBMS prototype, if a query references only formatted data, it is passed to INGRES directly, but, if a query includes media data then the query is decomposed into multiple subqueries. Each of the subqueries is individually processed, and the intermediate results are recomposed to form the final result to be given to the user. However, the early prototype

version did not support complex queries. This thesis will concentrate on complex queries such as nesting conditions and multiple selections.

This thesis is organized in six chapters and three appendices. The next chapter, Chapter II, gives a survey of previous work done in the MDBMS project. Chapter III reports the modularization of the MDBMS prototype program code. Chapter IV will give the design for complex query processing. Chapter V will present the implementation of complex query processing. Chapter VI will give the conclusion and summary along with a brief statement of other work planned or in progress. Appendix A will present a comprehensive example for the retrieval process using complex queries. Appendix B will give the generation of embedded SQL code for complex queries, and finally Appendix C will present the program code.

II. SURVEY OF PREVIOUS WORK

As mentioned in the previous chapter, multimedia data consists of media data such as image, text, voice, signals, etc. in addition to formatted data. A multimedia database management system (MDBMS) is defined as a system that manages all multimedia data and provide mechanisms to handle concurrency, consistency, and recovery in addition to providing a query language and query processing. In this chapter we present the data organization for multimedia objects, integration of conventional and multimedia data, architecture of the MDBMS prototype, natural language understanding capabilities in the parser which are required for the content retrieval of multimedia data, and finally the retrieval component of the prototype implemented so far.

A. DATA ORGANIZATION FOR MULTIMEDIA OBJECTS

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

The fundamental difficulty in handling multimedia data is intrinsically tied to a very rich semantics. To illustrate such a difficulty, let us look at an image of ships. Given such a picture, how are we to know what type of ships are in the picture. In other words, are the ships destroyers, cruisers, submarines or passenger ships? As another example, let us

suppose that we have a picture of soldiers. How do we know that the soldiers are fighting in a war or they are performing an exercise?

To answer queries posed on images, for example, a person must draw from a very rich experience encountered in life to derive a good answer. One must have a sophisticated technique to analyze the contents of the images to get the semantics of different things in the images. Today's technology does not allow systems to have this kind of capability to answer multimedia queries. However, we can use both Artificial Intelligence (AI) and Information Retrieval (IR) technology to do the next best thing. We can abstract the contents of the multimedia data into words or text and use the text description equivalent of the original multimedia data to match the user query. This is the principle used in the design of the MDBMS prototype to handle multimedia data for different applications. Figure 2.1 shows the format of image data and Figure 2.2 shows the format of sound data; both of them consist of the registration, raw and description data.

Raw data is the bit string representation of the image, sound, signal, etc. obtained from scanning or digitizing the original multimedia data. Registration data generally enhances the information about raw data and is not redundant. Description data describes the contents of the multimedia data and cannot be automatically derived by the computer with today's technology. The description data for multimedia data is to be supplied by users.

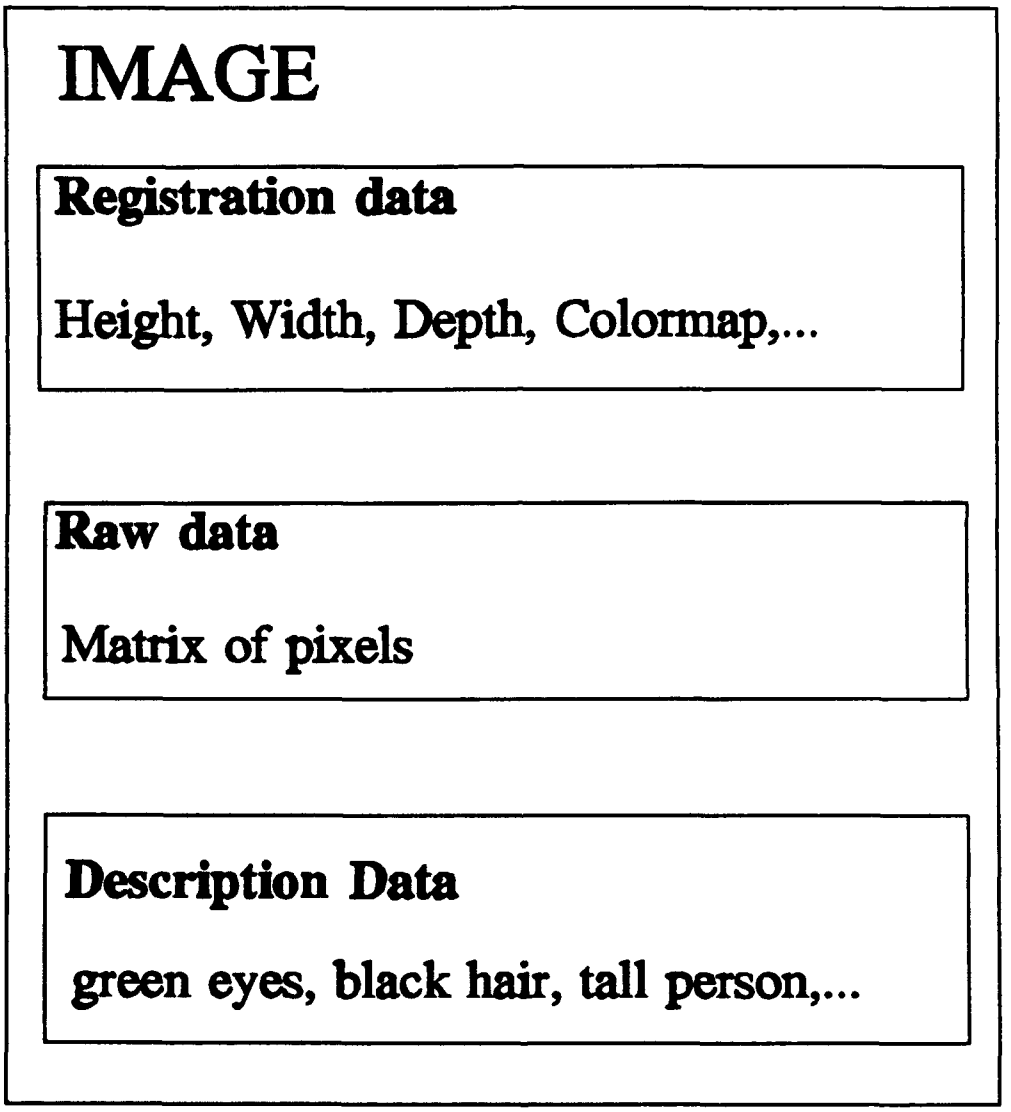


Figure 2.1 Structure of an Image Object

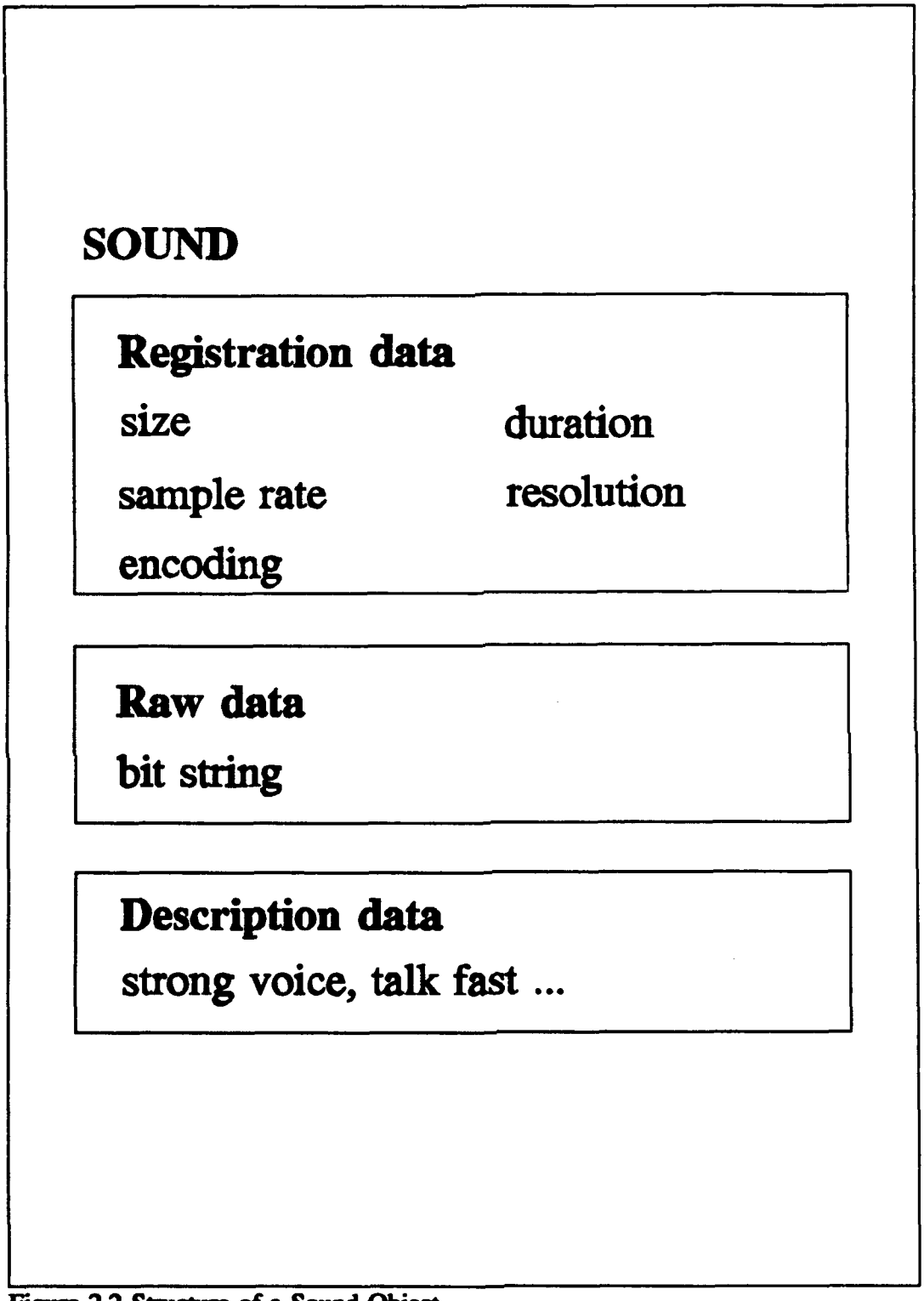


Figure 2.2 Structure of a Sound Object

B. INTEGRATION OF CONVENTIONAL AND MULTIMEDIA DBMS

The relational model has been selected as a basis to design and build the MDBMS prototype since the relational model is well known and widely used and has a firm theoretical basis.

When a relation has an attribute with a media type (i.e., data type of the attribute: sound or image), then an additional relation, called media relation, has to be created for storing registration and description data as shown in Figure 2.3. For each attribute with media data type a separate media relation is created.

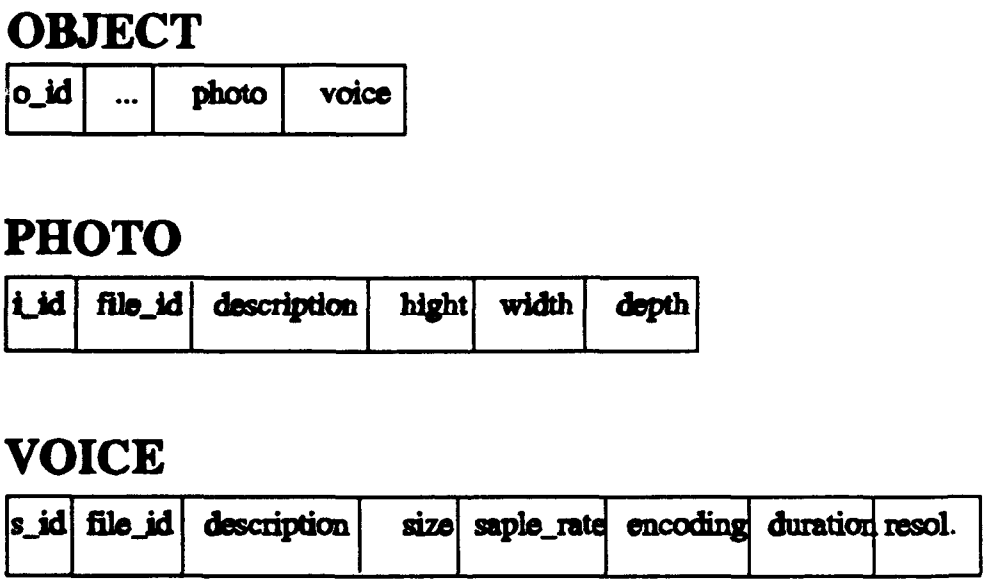


Figure 2.3 Schema for Modeling Relationship between Standard Objects and Media Objects

In Figure 2.3 the relation called OBJECT has the media attributes photo and voice in addition to other attributes with conventional data types. For the photo attribute a media relation is created and named after its attribute name, namely PHOTO. The PHOTO relation has i_id as the table key linking the PHOTO relation to the OBJECT relation. It also has the attributes file_id with the path to the file where the raw data for the image object is kept, description which is natural language description of the content of the image, and also height, width and depth which constitute the registration data part of the image object. In the same way, for the voice attribute of the OBJECT relation another media relation called VOICE is created. The VOICE relation has the attributes s_id as the table key, file_id showing the path to the file where the raw data for the sound object is kept, description describing the content of the sound object and finally size, sample_rate, encoding, duration and resolution as the registration data part of the sound object.

C. ARCHITECTURE OF THE MDBMS

In this section we will present various components of the MDBMS prototype. The components of the MDBMS are User Interface, Query Processor, Data Access Subsystem and Intelligent Retrieval Subsystem (See Figure 2.4).

The Data Access Subsystem consists of Conventional Data Manager and Media Data Manager and controls the access to the actual data stored in relational and media DBMS. The Intelligent Retrieval Subsystem is composed of Parser, Generator, Matcher

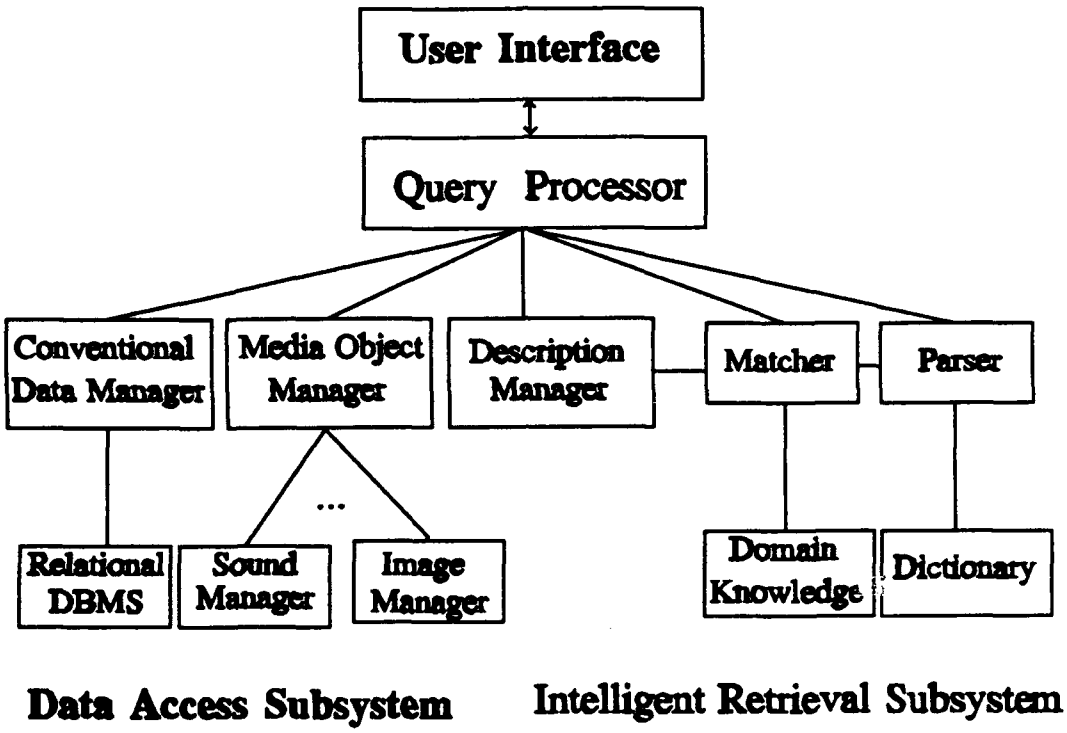


Figure 2.4 Architecture of the MDBMS Prototype

and Description Manager. The Query Processor accepts queries from the user and executes them by calling the other components. When a new description for a media data is entered, for example, the query processor calls the parser. The parser uses the dictionary to produce first-order predicates and return them to the query processor. The

query processor, then, hands the predicates over to the description manager which links the description to its multimedia data.

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

The conventional data manager, media object manager, description manager, parser, matcher and part of the query processor have already been implemented as part of the MDBMS prototype [REF6, REF7, REF8].

D. NATURAL LANGUAGE UNDERSTANDING IN THE PARSER

In this section we present the natural language understanding capabilities of the parser. In order to accomplish the goal of content retrieval of multimedia data, complete understanding of natural language is not necessary. However, a restricted interpretation

is necessary and this is done by the parser component using the application dependent dictionary as a semantic basis.

1. Natural Language Description for Multimedia Data

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

Additionally, for a particular multimedia application the universe of discourse is usually quite constrained. Nouns tend to be concrete and most multimedia databases emphasize still photographs and other fixed time graphics to which few verbs can be applied thereby easing a difficult aspect of natural language processing. The important thing is that we use natural language only to access entities in a database making complete understanding of all aspects of a word unnecessary.

2. Dictionary

Besides the captions themselves, the MDBMS prototype requires auxiliary information from a dictionary. The dictionary or lexicon is necessary for parsing and gives each possible natural language word its semantic: its part of speech, its grammatical

form and the form of literals needed to represent it. Many of the words for example, conjunctions and qualifying adjectives are consistent in meaning across a wide range of domains; thus we can borrow their interpretation from existing natural language systems and include them in every dictionary. The words that significantly change between applications are nouns and few verbs, need to be defined for every applications domain separately, but mostly their meaning is straightforward. To simplify matching, we try to limit the properties and relationships to a small set of primitives, for example we will not distinguish between the relationship asserted by the terms 'within', 'inside', 'part of', 'containing', 'including' and 'compromising'. This can be done without loss because in order to achieve efficient retrieval it is not necessary to capture the full meaning of an English expression, but just the main intent.

The dictionary is an important part of the system which is application dependent. In order to allow an interpretation of natural language captions it defines the domain of each application thus restricting their vocabulary, the semantics and the knowledge of the system to apply all the information.

3. Natural Language Interpretation

The parser translates the text description into a set of predicates called meaning list. The imprecision and ambiguity of the natural language descriptions is reduced considerably by transforming them into a set of predicates. These predicates state facts about the real world entities involved with multimedia data like their properties and relationships. As in most parsing methods, we chose first-order predicate calculus as a formal representation of the description data. The parser depends on the dictionary to turn

the descriptions into predicates. It is the parser's task to use the dictionary to resolve synonyms and to check the syntactic context to resolve lexical ambiguities.

Other important features of the parser are the use of supercaptions, a generalization of captions, and frames for stereotypical actions, allowing a set of predicates to be derived from terms in the description.

The current implementation of the parser uses augmented-transition network parsing and interpretation routines. It is implemented in Quintus Prolog and running on a SUN SPARC workstation. The details of the parser and the predicates are beyond the scope of this thesis and are given in [REF6, REF11].

An example of a natural language description and its translation into an equivalent set of predicates using the parser is shown below:

Description: "A cruiser with long_range missiles"

Predicates: ship(x), component(x,y), missiles(y), distance(y,long_range)

Choosing the right set of predicates is a very difficult task which is comparable to knowledge acquisition for expert systems. For the purposes of this thesis, it is sufficient to assume that the dictionary lists all the words the parser can recognize, all parts of speech associated with any word, and the predicates to use when a word appears in a description. Thus, the set of all predicates that can be used in the descriptions must be defined in the dictionary.

E. RETRIEVAL PROCESS

Retrieval is the most important operation in a MDBMS. In the existing MDBMS only simple SQL selections are implemented. As mentioned earlier, if a user query involves only formatted data then it is directly passed to INGRES. However, if a query includes media data then it is decomposed into multiple subqueries. Each of the subqueries is individually processed, their results are kept in temporary result tables and, finally, the results of all subqueries are recomposed to give the final result to the user. The existing system [REF10] did not support complex queries such as nesting conditions and multiple selections.

Nesting condition means one or more queries are placed inside another query. The inner query is called **subquery** and the encapsulating one is called **outer query**. The depth of nesting condition may be arbitrary according to the need.

Multiple selections can be presented in disjunctive normal form by using the Boolean operator '**and**' inside each group, and the Boolean operator '**or**' between groups. There may be one or more conditions inside each group and there may be one or more groups in a query.

The design and implementation of nesting conditions and multiple selections, which have not been supported by the MDBMS prototype so far, will be presented in Chapter IV and Chapter V of this thesis in detail.

III. MODULARIZATION

A. GENERAL

When the Multimedia Database Management System was growing and the number of people working on the system increased, we needed a better way of structuring the system. We decided that the best way was to divide the MDBMS program code into smaller units and get each person to work on his part separately without interfering with other people's parts. In this chapter we will first present the general concepts of modularization. Then, we will show how we used the C programming language to implement the MDBMS prototype to achieve modularization. Since modularization of the MDBMS prototype was a team effort, this chapter will also be included in [REF12].

1. What is Modularization?

Modularization is the process of structuring programs (i.e. data structures and functions) by dividing them into smaller units called modules. A module is a collection of related programming language entities (procedures, types, and so on). The different modules of a program are in relationship with each other resulting in a module hierarchy. Modules on a higher level of the hierarchy use functions or procedures of lower level modules.

2. Why do we need Modularization?

A program which does not consist of substructures is hard to understand. Dividing a program into modules makes it easier to follow. Division of labor among

people shortens the time to complete a project. When need arises to change the program code, changing one or a few modules will be enough to get the desired result. When a new system is to be built, it is possible to reuse modules of a previous system. Testing of modules is easier than testing a program code without any structure. In the following part we will discuss the advantages of modularization in detail.

a. Comprehensibility

A system with no substructure is hard to understand. Years of experience shows that modules with high cohesion and low coupling supply designers with easier-to-understand systems. High cohesion means that the functions and objects within a module are closely related to each other. Low coupling means that each module interacts with some others through a narrow interface.

b. Division of Labor

To complete a large task in a reasonable time, it must be divided among the people participating the project. This can be done using modules as basic units. Each module given to a person of the project should be small enough to be implemented within a relatively short period of time. If the implementation of a module would take too long, then it is necessary to break it into smaller pieces.

c. Response to change

Change is a fundamental characteristic of software systems. Users may ask new features or changes to old ones; the system may move to new hardware or a new operating system; bugs may be discovered during testing; performance measurements may

show bottlenecks. All these changes done to an existing software system cost large amounts of money. Modularization makes it easier to do changes. Changing a module, instead of changing the whole system gives the desired result.

d. **Reusability**

When you build a new system, you may need the same functionality that you used in a previous system. For instance, a module for string utility functions or I/O functions will be useful for many applications. Instead of rewriting these modules, it saves much effort if you can reuse modules of a previous system.

e. **Easier to test**

Smaller parts are easier to debug. In a large system each module should be tested individually, and large collections of modules should be slowly built up to test the whole system. Debugging is the search for defects; it often involves a lot of detective work. Testing has much broader scope, and usually assumes you have finished most of the debugging. Testing may uncover problems, which may lead to further debugging.

In addition to these advantages, modularization can be used to achieve two basic principles in systems development: information hiding and abstraction (or encapsulation). The principle of information hiding is that each module hides some design decision. If the design decision changes, then only the module hiding the design decision need to be changed. All other modules using the changed module do not need to be changed. Information hiding and abstraction focus on different aspects. Information hiding

focuses on what to hide; abstraction focuses on what to reveal. Information hiding tries to protect you from change; you ask what design decisions might change, and arrange to hide them so you cannot depend on them. The sorts of decisions one might hide include:

- The algorithm for carrying out some operation.
- The representation of some data structures.
- The details of an interface to an operating system, or to special purpose hardware.
- The policy for allocating some resource or ordering certain operations.

Every abstract data type is an information hiding module; however a module may not be an abstract data type. A module can also be a set of unrelated functions. An abstract data type module provides a collection of procedures for manipulating the encapsulated data structure. For example, a module might hide the representation of a stack. It would provide operations for pushing elements onto a stack, popping elements off the stack, reading the top element of the stack and initializing the stack. These four functions are called the interface procedures of the module.

3. How to modularize

Now, we need to address how to achieve a modular system. In decomposing a system into modules five phases are necessary:

a. Identify major groups of design decisions, subgroups within those groups, and so on. These become the higher levels of the module hierarchy, and chapters, sections and so forth of the decomposition document. The decomposition document records the division of the system into modules. It is used as a baseline document for detailed design.

b. Identify all the major design decisions in your project, and record modules that hide them in the decomposition document. These become the leaves of the module hierarchy.

c. Estimate the size of each module. If it seems too large for one person to handle, break it into smaller pieces.

d. The results of the previous phase are separate modules. Now, the dependencies between modules need to be specified. A module dependency document defines the module dependencies and describes a module hierarchy. Figure 3.1 shows how a module hierarchy might look like. A module hierarchy reflects the structure of the whole system. Each box represents a module and each line connecting two modules

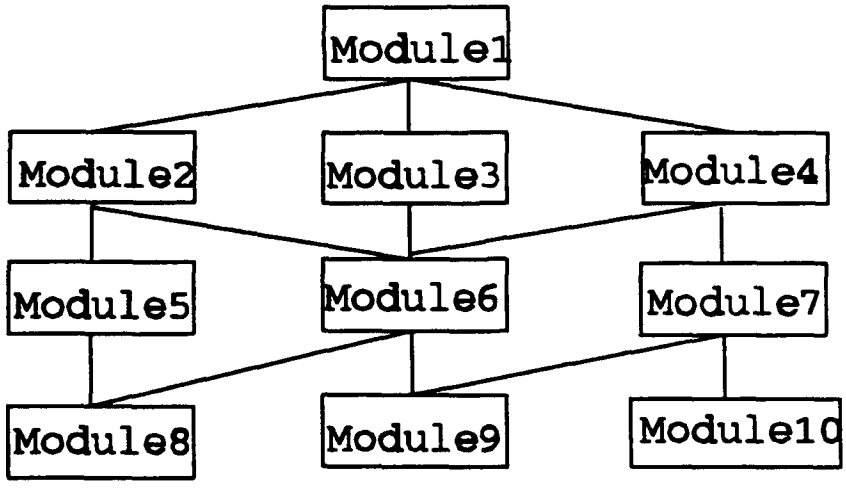


Figure 3.1. An Example Module Hierarchy

represent a dependency between modules. For example in Figure 3.1 Module 1 calls modules Module 2, Module 3 and Module 4. Module 1 is at the first level of the hierarchy while Module 2, Module 3 and Module 4 are at the second level of the module hierarchy and so on.

e. The last phase is to identify for each module the relationship with other modules. To make the dependencies between modules, import and export interfaces are added to the documentation header of each module. In the import interface of each module, functions that are called from other modules are placed. The export interface summarizes the functionality provided by a module. In the export interface of each module, functions that can be used by other modules take place. Each module in the module hierarchy is structured as follows (Figure 3.2):

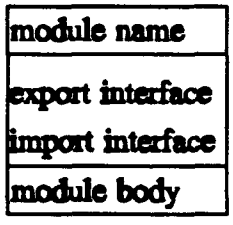


Figure 3.2 Structure of a Module

export interface:=export <function 1>,<function2>,...

import interface:=import <function 1>, <function2>,...

from module <module 1>

import <function 1>, <function 2>,...

from <module 2>

...

module body:= implementation of the exported functions by using the imported functions (including not exported functions and data structures)

B. MODULARIZATION OF C PROGRAMS

C programming language (Kernighan and Ritchie C) was chosen to implement MDBMS when the studies began on the prototype in 1988. However, C does not support modularization. The only thing that can be done is to divide the program code in parts and store them in separate files. This allows the user to use separate compilation or the include mechanism provided by the C language.

Files were not designed as mechanism for information hiding and data abstraction. They were provided as a facility to support program partitioning and independent compilation. Files containing components of a C program (functions, declarations and definitions) can be compiled independently. Independently compiled program components, along with precompiled library functions, can be linked together to produce a complete program.

Since C files can be compiled independently, it is convenient to partition large C programs into smaller and more manageable parts to achieve some advantages of the modularization concept. Independent compilation allows files containing C program components to be checked separately for syntactic and semantic errors. Moreover, when

a program is modified, it is only necessary to recompile the effected components. The Unix utility 'make' is used to automate this process.

C files can be used to partly implement data abstraction and information hiding. An abstract data object, as defined in the previous section, is an object that can be manipulated using only the operations supplied by the definer of the object. The user cannot directly manipulate the underlying implementation of an abstract data object. Details of how an abstract data object is implemented are hidden from the user. Hiding the details prevents the user from:

- making programs dependent on the representation. The representation of an abstract data type can be changed without effecting the rest of the program. For example, the abstract data type set may be initially implemented as an array, but this representation may be changed to an ordered list later on for storage efficiency.

- accidentally or maliciously violating the integrity of an abstract data type object. Integrity of abstract data type objects is preserved by forcing the user to manipulate these objects using only the operations provided by the designer of the abstract data type.

Examples of abstract data types are stacks, queues, sets, databases and binary trees.

However, a C file is not a true data abstraction facility, because it only partially supports data abstraction. If you link an independently compiled C module to some other parts, you can not prevent the user from accessing all functions and even the internal data structures of other modules.

The other mechanism used to achieve some kind of modularization is the include mechanism. Arbitrary files can be textually included in a C program by means of the include instruction. The capability to include files textually in a program allows common constant, data, type and function declarations and definitions to be kept in

separate files. These common declarations and definitions can then be used in all parts of the program. Keeping common declarations and definitions in separate files and then including them in C programs is a popular style used for writing C programs. A common example is the standard input/output declaration file stdio.h.

As mentioned before, although the C language does not support modularization we can achieve the following advantages by dividing the existing program code into separate parts:

- Programs modules can be developed independently.
- Changes to the program can be done by only changing single modules.
- Clarity of design and structure.
- Program code is easier to understand.
- Maintainability.
- Reusability of modules.
- Uniformity.

C. MODULARIZATION OF MDBMS

As mentioned in the previous section, the C language was chosen to implement the MDBMS prototype. When we started to implement the complex query processing, the program code was mainly in one file. Considering the size of the program and that multiple students are working on three different parts of the project, namely complex query processing, graphical user interface, modify and delete, we decided to modularize the MDBMS prototype.

Before starting the modularization, the module hierarchy of the MDBMS program code was as shown in Figure 3.3. First of all we divided the existing program code in separate files corresponding to their purpose. Then we divided each part again until we obtained the final module hierarchy (Figure 3.4).

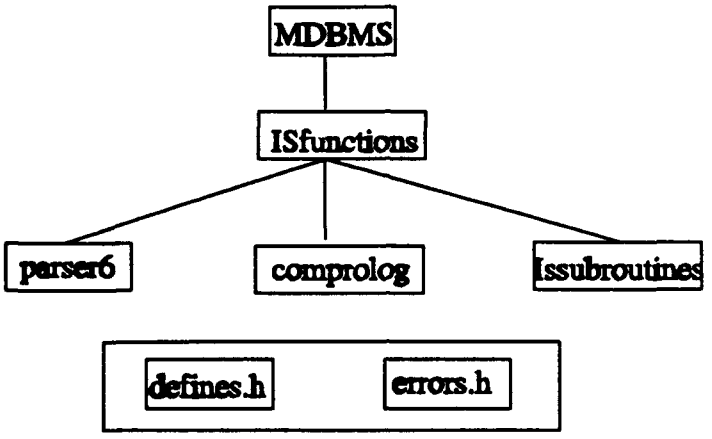


Figure 3.3 Module Hierarchy of MDBMS at the beginning

The MDBMS module hierarchy now consists of 6 levels. In Figure 3.4, each box represents a module. For instance, the module 'MDBMS' which is the main program calls the modules 'Catalog Management', 'Create', 'Insert', 'Modify', 'Delete', 'Retrieve' and 'Connect'. In the diagram the straight lines show the dependencies between modules. Although the module 'MDBMS'which is at the first level of the hierarchy calls the module 'Retrieve' which is at the third level of the hierarchy, the dependency is not shown on the diagram in order not to further complicate the module hierarchy diagram.

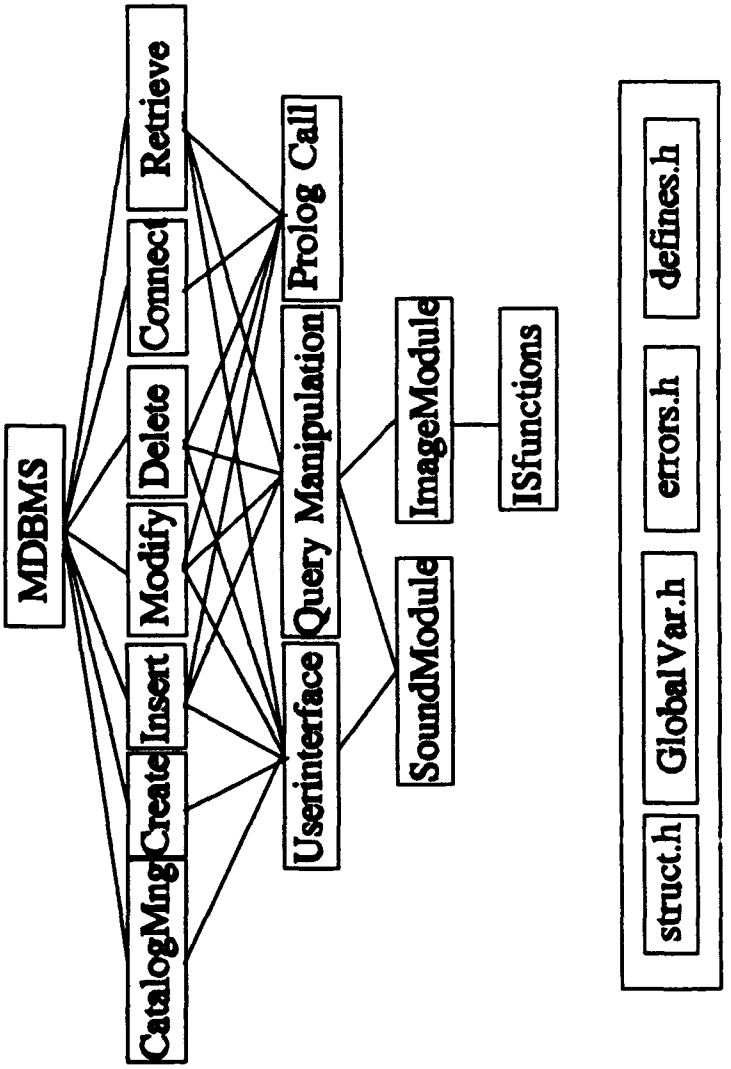


Figure 3.4 Module Hierarchy of MDBMS

There is, however, still much to do on modularization. Considering the time we needed to complete our part (i.e., complex query processing) we stopped working on modularization at this point.

To make the dependencies between the different parts of the MDBMS program visible, we have added import and export interfaces in the documentation header of each module (Figure 3.5).

In the export interface of each module, functions that can be used by other modules are placed. The export interface therefore summarizes the functionality provided by a module. For example, the function **print_all_table()** taking place in the export interface of 'Insert' Module can be called by the 'Retrieve' Module.

In the import interface of each module, functions that are called from other modules are mentioned. For instance, the function **get_sound_value()** taking place in the import interface of 'Insert' Module, is called from the 'User Interface' Module.

In addition to our work on modularization, we also used some helpful tools provided by the UNIX system, namely sccs, lint, make, and dbx. Sccs is a version manager which allows us to have more than version of a file. If you want to try another way to implement a module, you can work on it that while you still keep the older versions. You can always go back and work on any older version you want. The first thing in order to use sccs is to make a directory and call it SCCS. To create the first version of any file type "sccs create <filename>" at shell prompt. If you want to keep a version and also try something on the same version type "sccs delget <filename>". At this point you would have no writable copy of the file. To see how many versions of a file

```
*****************************************************************************
Title          : InsertModule.c
Author         : Su Cheng Pei
Date           : November 15, 1990
History        :
Description    : This module implements the insertion process
                 in the Multimedia Database System.
*****************************************************************************
Export Interface    :
     print_all_table()     :Prints out the table catalog information on the
                            screen.
     insert_tuple()        :Inserts a tuple of a particular relation.
     display_tuple()       :Displays the tuple before insertion.
     check_media_description():Checks the media description by connecting
                            to the parser.
     ql_insert_tuple()     :Translates SQL statement to insert a standard tuple.
*****************************************************************************
Import Interface    :
     get_sound_value()  :Gets a sound value of a media attribute from the
                         user input.
     clr_scr()          :Clears the screen.
     yes_no_answer()    :Gets yes or no answer from the user.
                         from UserInterface.c
     check_table_name():Checks if the table name is duplicate.
     get_media_name()   :Gets media table name by appending table_key at
                         the end of att_name.
                         from CreateModule.c
*****************************************************************************
```

Figure 3.5 Export / Import Interface of a Module

you have, type "sccs prs <filename>". To select a version and have a writable copy of a file, type "sccs edit -r<version number> <filename>" (for instance if we want to edit version number 1.1.12.4 of the file RetrieveModule.c, we should type "sccs edit -r1.1.12.3 RetrieveModule.c"). To see which versions of all files are currently being edited, type "sccs info" only. These are the main commands for using sccs.

The second tool we used was lint. During the compilation of C programs lint helps to find:

- unused arguments,
- unused variables,
- variables which are set but not used,
- inconsistently used function calls,
- always ignored function returns,...

We put the lint command in the Makefile so during compilation lint is invoked automatically by the Makefile. See Chapter V.C of this thesis to see how lint is used in the Makefile.

The third utility program we used was 'make' which is a command generator. Refer to V.C of this thesis to get detailed information about 'make' and its use.

The last tool we used was dbx. Dbx is a source-level debugger. It helps find run time errors during execution of a program. To run dbx, type "dbx <executable file name>". To run the program, type "run". To trace in a function, type "trace in <function name>.

IV. DESIGN OF COMPLEX QUERY PROCESSING

In Chapter II of this thesis the general architecture of the MDBMS prototype is described in detail. Basically, it was an attempt to broaden the database handling capability by providing the integrated support of both formatted and media data. The design of complex query processing is done based on the architecture presented. However, several resource constraints in INGRES, the IBM compatible PC and the SUN workstation were found when studies started on the MDBMS and these restrictions influenced the design and implementation of the prototype. In this chapter we mention the system environment and give a sample application. Further, we present the design of complex query processing in detail.

A. SYSTEM ENVIRONMENT AND SAMPLE APPLICATION

1. System Environment

The MDBMS prototype was built on top of INGRES to support formatted and multimedia data. INGRES acts as the manager for the data storage. However, INGRES has a lot of restrictions:

- INGRES does not support ADT, the approach selected to support multimedia data.
- INGRES does not allow its users to get the catalog information readily.
- Although INGRES supports embedded SQL in host C language, it does not provide a set of high level function calls available to the users. For example, the embedded SQL statements are pre-compiled into INGRES low level code for execution. It does not allow the relation name and attribute name as a program variable in the high level embedded SQL code.

- INGRES does not support set operations such as UNION, INTERSECTION and MINUS (i.e., difference of two tables). UNION and INTERSECTION are of great importance to be able to design and implement complex queries.

Although more recent versions of INGRES have removed some of these restrictions, a significant recoding effort would be required for using the new version. However, some coding effort had to be done, as we will also mention later in this chapter, to extend the capabilities of the INGRES SQL for supporting the set operations UNION, INTERSECTION and MINUS.

In the meantime, a similar situation occurs in the SUN workstation. New SUN workstations now support sound, but it would require a substantial investment to purchase new hardware and recode the prototype source code. It was decided that instead of these investments, the PC would be retained to manage sound data and would be incorporated into MDBMS prototype as a backend server by connecting it to the SUN system via a local network, i.e., ETHERNET [REF1].

Similarly, to capture images, a video card which works with a camcorder is installed into another PC. The PC first captures an image in GIF format. This file is then transferred to the SUN workstation using FTP (File Transfer Protocol) in binary mode. The image files in GIF format are transformed into RASTER format by software before they can be used by the MDBMS prototype. A more detailed description of the capturing process of the images is described in [REF10].

All of these constraints affected the design and implementation of the MDBMS prototype. Since the prototype construction is not intended to be a production system at

this time, and because the current system is enough to demonstrate the principles, a decision was made not to change the structure of the system.

2. Sample Application

Many application areas increasingly require a MDBMS to manage both formatted and multimedia data. Examples can be found in military, publishing, entertainment and instructional environments. In this subsection, we present a sample application which can be considered quite typical in a military environment. The goal is to give the reader a better understanding in the design and implementation of complex queries for multimedia processing.

Let us assume that the Chief of the Navy has ordered his staff to keep information about Navy ships, weapons, officers, missions of the ships and bases of the ships. Suppose we want to store in the database the names, types, ID's, displacements, mission id's and base id's of the ships, the years in which the ships are built, the captains and executive officers of the ships, and finally the pictures of the ships. Let us assume that we want to know what weapons are on the ships and the weapons' power, fire range and the weapons' pictures. As for the officers, their names, ranks, ID's, salaries, report dates as well as their pictures and voices should also be kept in the database. Moreover, we may want to keep in the database the name, direction, goal and task related to each mission and also the name, location, and size of each Navy base. As seen from this example, besides standard data types we also have media types, namely image and sound. The above information can be transformed into relations in a database as shown in Figure 4.1.

SHIP

s_name	s_no	type	yr_built	disp	m_id	b_id	capt_id	exo_id	picture

SHIP_WEAPON

s_no	w_name

WEAPON

w_name	type	fire_range	power	picture

OFFICER

o_id	o_name	rank	salary	rep_yr	picture	voice

MISSION

m_id	m_name	direction	goal	task

BASE

b_id	b_name	location	size

Figure 4.1. Navy Ship Relational Database Schemes

The primary keys (underlined) of the relational schemes in Figure 4.1 are externally defined by the MDBMS user, and the media data types such as image and sound have also been defined as data types supported by the MDBMS prototype. As mentioned earlier, INGRES is used to store all the data. The question now is how to store media data types in INGRES which supports only standard data types? The solution is to express media data types in terms of standard data types. In the MDBMS prototype, the data type of each media attribute is defined as INTEGER internally. The content of

the media type is an integer which link to its own media relation that is not transparent to the users. These integers are internally generated identifiers for the tuples in the media relations. For each media attribute, a media relation is generated. This is deemed desirable since putting media data together, i.e., images from different relations, does not produce any benefit but actually causes the system to degrade in performance. Hence, "picture" in the relation OFFICER requires a media relation and picture in WEAPON requires another. Since attribute names do not have to be unique across relations, we must find ways to name the two PICTURE relations differently. The solution is to append the relation's internal identifier to the media attribute names. Let us assume the SHIP's internal identifier is "1", then the image media relation for the attribute "picture" in SHIP becomes PICTURE1. In the same way, to each media table is assigned a name resulting in the media relations' names as shown in Figure 4.2. Note that all the media tables are invisible to the users.

Given the sample application above, the MDBMS prototype before the design and implementation of complex queries was able to respond queries as follows:

- Retrieve the picture and voice recording of the captain of the ship "Kitty Hawk"?
- What are the names of the ship weapons whose fire range is greater than 200 miles and whose pictures show "long range missile against land targets"?
- Which ship is the executive officer Rosemary Stewart stationed at?
- Retrieve the pictures and names of the ships which have the weapon "Trident".

PICTURE1

i_id	f_id	descrp	height	width	depth
(int)	(c64)	(vc500)	(int)	(int)	(int)

PICTURE3

i_id	f_id	descrp	height	width	depth
(int)	(c64)	(vc500)	(int)	(int)	(int)

PHOTO6

i_id	f_id	descrp	height	width	depth
(int)	(c64)	(vc500)	(int)	(int)	(int)

VOICE6

s_id	f_id	descrp	size	samp_rate	encoding	duration	resolution
(int)	(c64)	(vc500)	(int)	(int)	(int)	(float)	(int)

Figure 4.2. **The Media Relational Database Schemes for Media Attributes in Figure 4.1**

However, the prototype could not respond the types of queries listed below:

- Retrieve the pictures, voice recordings and names the captains of the ships "Kitty Hawk" or "Mississippi"?

- List the name of all ships which have the weapon whose picture shows "high speed guided torpedo" and whose fire range is greater than 1 mile or which was commanded by Captain "Huseyin Aygun".

- Retrieve the voice recordings and names of the officers whose salary is greater than 30,000 but who are not captain.

Now let us look at how these queries can be handled in the order they are given:

- Can be handled using multiple selections.
- Can be handled using a nested query inside multiple selections.
- Can be handled using set operation MINUS.

The detail information about how these queries and other complex queries can be evaluated will be given in the next section.

B. DESIGN OF COMPLEX QUERY PROCESSING

In this section we will first review the design of simple queries [REF10], then present the design of complex queries, namely nesting conditions (i.e.,IN, NOT IN, EXISTS, NOT EXISTS) and multiple selections, along with the design of set operations (i.e., UNION, INTERSECT, MINUS) and aggregate functions. Finally give an example for multiple selections including nesting condition. With our approach for the design of complex queries, it is possible to have nested queries up to arbitrary depth and arbitrarily many conditions inside each group and arbitrarily many groups for multiple selections.

1. Simple Queries

In this subsection we review simple queries which have already been designed and implemented by [REF10]. Further we will point out the differences of simple queries between our approach and the approach in [REF10].

As mentioned earlier, if a query includes only formatted data, it is directly passed to INGRES. However, if a query includes media data then the query is decomposed into multiple subqueries. Each subquery is then individually processed and the results of these subqueries are recomposed to give the final result to the user. Now

we can look at two examples - one with only formatted data, the other including media data in addition to formatted data - to illustrate what we have just said.

1. Query: Which Navy ship is "Rosemary Stewart" stationed at? The SQL statement for this query can be written as follows:

```
SELECT  s_name
FROM    ship, officer
WHERE   ship.exo_id=officer.o_id and o_name="Rosemary Stewart"
```

Since this query contains only formatted data, no decomposition is necessary and it is directly passed to INGRES to get the result.

2. Query: Retrieve the names, pictures and voice recordings of the executive officers stationed at ships whose displacement is greater than 40,000 and whose picture shows "gas turbine powered ship"? The extended SQL statement for this query can be written as follows:

```
SELECT  o_name, picture, voice
FROM    officer, ship
WHERE   officer.o_id=ship.exo_id and displacement > 40,000 and ship.picture
(CONTAINS, "gas turbine powered ship");
```

Since the above query contains media it should be decomposed into subqueries. The decomposition process is shown below:

Create table T1 as :

```
    SELECT *
    FROM    ship, officer
```

WHERE officer.o_id=ship.exo_id and displacement > 40,000;

Create table M1 as :

SELECT i_id

FROM PICTURE1

WHERE PICTURE1 (CONTAINS, "gas turbine powered ship");

Create table RESULT as :

SELECT o_name, picture, voice

FROM T1, M1

WHERE T1.picture=M1.i_id

After the system gets the final result which is an INGRES relation, the system will generate a cursor called **cursor_output** to print out the data one tuple at a time. If the output contains any media data, as in the above example, the RESULT table shows us the tuple id's retrieved from the related media relation, in the example above the media relation is PICTURE1. Later the system displays the media data in the order printed out for the formatted part. The process of creating and using a cursor is as follows:

EXEC SQL CREATE CURSOR cursor_output AS

SELECT *

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 pictures;

play voice recordings;

What has been discussed about simple queries so far is according to the approach in [REF10]. We found that it is not convenient to display the media data in the order printed out for the formatted data. What if the user wants to see only the picture of the last tuple displayed as the final result? So, we modified the design for the display process of media part. With this modification, the user of the MDBMS prototype can select which media data to display. More detailed information about the modification can be found in the next chapter of this thesis.

Because the design for the process of decomposition, when a query includes media data, introduced in this subsection is the same for complex queries and set operations which will be introduced in the rest of this chapter, we will not repeat the decomposition process due to the inclusion of media data in a query for the sake of clarity.

2. Nested Queries

Some queries require that existing values in the database be fetched and used in a comparison condition. Such queries can be conveniently formulated using **nested queries** which are complete SELECT- FROM-WHERE queries within the WHERE clause of another query which is called the **outer query**. In this chapter we present the design of

nested queries including the comparison operators IN, NOT IN, EXISTS, and NOT EXISTS giving examples for each to clarify the approach.

a. IN:

The comparison operator IN compares a value, say v, with a set (or multiset) of values V and evaluates to TRUE if v is one of the elements in V. Let us now give an example to show how a nested query with the comparison operator IN looks like.

Query: Retrieve the names, pictures and voice recordings of all executive officers stationed at ships whose weapon's picture shows "high speed guided torpedo".

An extended SQL statement for the above query:

```
SELECT o_name, picture, voice
FROM officer
WHERE o_id IN
    (SELECT exo_id
     FROM ship
     WHERE s_no IN
        (SELECT s_no
         FROM ship_weapon, weapon
         WHERE ship_weapon.w_name=weapon.w_name and
              weapon.picture(CONTAINS, "high speed guided
              torpedo")));
```

This query is evaluated as follows:

Create table T1 as:

SELECT s_no

FROM ship_weapon, weapon

WHERE ship_weapon.w_name=weapon.w_name and

weapon.picture(CONTAINS, "high speed guided torpedo")));

Create table T2 as:

SELECT exo_id

FROM ship, T1

WHERE ship.s_no=T1.s_no

Create table RESULT as:

SELECT o_name, picture, voice

FROM officer, T2

WHERE officer.o_id=T2.exo_id

Note that we have neither shown the creation of temporary media tables nor the display of the final result as we did in the previous section for simple queries. The idea is to emphasize the design of a nested query with the comparison operator IN. The same process will be followed for the rest of this chapter.

b. NOT IN:

The comparison operator NOT IN compares a value v with a set (or multiset) of values V and evaluates to TRUE if v is not one of the elements in V. An example of a nested query with the comparison operator NOT IN is given below:

Query: Retrieve the names, displacements, and pictures of all ships which do not have weapons whose picture shows "long range underwater-to-surface missile".

An extended SQL statement for the above query can be written as follows:

```
SELECT s_name, displacement, picture
FROM ship
WHERE s_no NOT IN
    (SELECT s_no
     FROM ship_weapon, weapon
     WHERE ship_weapon.w_name=weapon.w_name and
         weapon.picture(CONTAINS, "long range underwater-to-surface
         missile").
```

The above query is evaluated as follows:

Create table T1 as:

```
SELECT s_no
FROM ship_weapon, weapon
WHERE ship_weapon.w_name=weapon.w_name and
    weapon.picture(CONTAINS, "long range underwater-to-surface
    missile").
```

Create table RESULT as:

```
SELECT s_name, displacement, picture
FROM ship, T1
WHERE (ship.s_no=T1.s_no)=FALSE
```

In the creation of the RESULT table the statement WHERE (ship.s_no=T1.s_no)=FALSE is used to show that the tuples of the table SHIP are retrieved into the table RESULT if the join condition evaluates to FALSE. In other words tuples of the table SHIP are retrieved if they are not a member of the values in table T1.

c. **EXISTS:**

The comparison operator EXISTS is usually used in conjunction with a correlated nested query. A correlated nested query is a nested query with a join condition related to the outer query. Nested queries with the comparison operator EXISTS work as follows:

For each tuple of the outer query, the nested query is evaluated; if at least one tuple exists in the result of the nested query then that tuple of the outer query is retrieved.

Query: Retrieve the names, ranks and pictures of the captains who commanded the ships whose pictures show "a cruiser firing at the enemy at the Gulf War".

SQL statement for the above query:

```
SELECT o_name, rank, picture
FROM officer
WHERE EXISTS
    (SELECT *
     FROM ship
     WHERE ship.picture(CONTAINS, "a cruiser firing at the
```

enemy at the Gulf War"));

The above query is evaluated as follows:

Create table T1 as follows:

SELECT *

FROM ship

WHERE ship.picture(CONTAINS, "a cruiser firing at the

enemy at the Gulf War"));

Create table RESULT as follows:

SELECT o_name, rank, picture

FROM officer, T1

WHERE officer.o_id=T1.exo_id

d. **NOT EXISTS:**

The comparison operator NOT EXISTS is also used in conjunction with a correlated nested query. NOT EXISTS works as follows:

For each tuple of the outer query, the nested query is evaluated; if the tuple does not exist in the result of the nested query then that tuple of the outer query is retrieved.

Query: Retrieve the names, ranks and pictures of the executive officers who are not stationed at destroyers.

SQL statement:

SELECT o_name, rank, picture

FROM officer

```
WHERE NOT EXISTS
    (SELECT *
     FROM ship
     WHERE type="destroyer");
```

The query is evaluated as follows:

Create table T1 as:

```
SELECT *
FROM ship
WHERE type="destroyer"
```

Create table RESULT as:

```
SELECT o_name, rank, picture
FROM officer, T1
WHERE (officer.o_id=T1.exo_id)=FALSE;
```

3. Set Operations

Set operations in SQL are INTERSECTION, UNION, and MINUS (set difference). As we mentioned at the beginning of this chapter, our INGRES version does not support any of these operations. Since set operations are of great importance to us for implementing complex queries, we extended the capabilities of the SQL by implementing the set operations in C. Below we present the design of set operations.

a. UNION:

The union of two tables is a table containing all rows that are either in the first, or in the second table or in both of them. There is an obvious restriction on this operation. It does not make sense, for example, to talk about the union of the SHIP and the OFFICER table. What would rows in this union look like? The two tables must have the same structure, i.e., they must be **union-compatible**. Two tables are union-compatible if they have the same number of columns and if their corresponding columns have identical data types and lengths. Note that the definition does not state that the column headings (attribute names) of the two tables must be identical but rather that the columns must be of the same type; thus if one is integer, the other one must also be an integer.

Our design for UNION is to retrieve all the tuples of the second table and insert them into the first table. This is considered to be the easiest way to implement the set operation UNION. Let us give an example to make our design clearer.

Query: Retrieve the names, ranks, pictures and voice recordings of all the officers who worked as an executive officer or as a captain on the ships whose pictures show "nuclear submarine with many kinds of guided torpedoes".

Extended SQL statement for this query:

```
(SELECTo_name, rank, picture, voice
FROM    officer, ship
WHERE officer.o_id=ship.exo_id and  ship.picture(CONTAINS,
            "nuclear submarine with many kinds of guided
            torpedoes"))
```

UNION

(SELECTo_name, rank, picture, voice

FROM officer, ship

WHERE officer.o_id=ship.capt_id and ship.picture(CONTAINS,

"nuclear submarine with many kinds of guided

torpedoes"));

The above query is evaluated as follows:

Create table T1 as:

SELECT o_name, rank, picture, voice

FROM officer, ship

WHERE officer.o_id=ship.exo_id and ship.picture(CONTAINS,

"nuclear submarine with many kinds of guided

torpedoes");

Create table T2 as:

SELECT o_name, rank, picture, voice

FROM officer, ship

WHERE officer.o_id=ship.capt_id and ship.picture(CONTAINS,

"nuclear submarine with many kinds of guided

torpedoes");

EXEC SQL CREATE CURSOR cursor_output1 AS:

SELECT *

FROM T1

EXEC SQL CREATE CURSOR cursor_output2 AS:

SELECT *

FROM T2

INSERT INTO table T1

VALUES (EXEC SQL FETCH CURSOR cursor_output2);

b. **INTERSECTION:**

The intersection of two tables is a table containing all rows that are in both tables. We should keep in mind that the issue of union_compatibility is also valid for intersection.

Our design for the intersection of two tables dictates that the two tables should be joined with all the column headings (attributes). This approach gives the same result as the approach in which each tuple of the first table is checked against all the tuples of the second table. Let us illustrate our approach with an example.

Query: Retrieve the names, power and pictures of all weapons whose pictures show "high speed close range defense weapon" along with the ones located on board the ship "Elliott".

Extended SQL statement for the above query can be written as follows:

(SELECTw_name, power, picture

FROM weapon

WHERE weapon.picture(CONTAINS, "high speed close range

defense weapon"))

INTERSECT

(SELECTw_name, power, picture

FROM ship, ship_weapon, weapon

WHERE ship.s_no=ship_weapon.s_no and

ship_weapon.w_name=weapon.w_name);

The above query is evaluated as follows:

Create table T1 as:

SELECT w_name, power, picture

FROM weapon

WHERE weapon.picture(CONTAINS, "high speed close range

defense weapon");

Create table T2 as:

SELECT w_name, power, picture

FROM ship, ship_weapon, weapon

WHERE ship.s_no=ship_weapon.s_no and

ship_weapon.w_name=weapon.w_name;

Create table RESULT as:

SELECT w_name, power, picture

FROM T1, T2

WHERE T1.w_name=T2.w_name and T1.power=T2.power and

T1.picture=T2.picture

c. **MINUS:**

The difference of two tables T1 and T2 (referred to as T1 MINUS T2) is the set of all rows that are in T1 but not in T2. Our design for the difference of two tables indicates that all the rows from the first table should be retrieved if the result of joining the two tables with all their attributes evaluates to FALSE.

Let us clarify this approach with an example:

Query: Retrieve the names, ranks, and pictures of all the officers whose picture show "tall person" but not the ones whose pictures show "blond hair".

Extended SQL statement for this query can be written as follows:

```
(SELECT o_name, rank, picture
FROM officer
WHERE officer.picture(CONTAINS, "tall person"))
MINUS
(SELECT o_name, rank, picture
FROM officer
WHERE officer.picture(CONTAINS, "blond hair"))
```

This query can be evaluated as follows:

Create T1 as:

```
SELECT o_name, rank, picture
FROM officer
WHERE officer.picture(CONTAINS, "tall person");
```

Create T2 as:

SELECT o_name, rank, picture

FROM officer

WHERE officer.picture(CONTAINS, "blond hair");

Create RESULT as:

SELECT o_name, rank, picture

FROM T1, T2

WHERE (T1.o_name=T2.o_name and T1.rank=T2.rank and

T1.picture=T2.picture)=FALSE

The clause "WHERE (T1.o_name=T2.o_name and T1.rank = T2.rank and T1.picture=T2.picture)=FALSE" means the tuples from T1 are retrieved if the join conditions evaluate to FALSE, in other words if those tuples are not in T2.

4. Aggregate Functions

Since aggregation is required in many database applications, we decided to implement the aggregate functions in addition to the complex queries mentioned in this chapter.

Aggregate functions like COUNT, SUM, MAX, MIN and AVG are built-in functions in INGRES SQL. In this subsection we present the design of the aggregate functions for our MDBMS prototype system.

a. COUNT

The built-in function COUNT returns the number of tuples found for a specified condition. Let us give an example to clarify how COUNT works:

Query: How many executive officers are there in the fleet, whose pictures show "tall person with black hair"?

The extended SQL statement for the above query:

SELECT COUNT(o_name)

FROM officer

WHERE officer.picture (CONTAINS, "tall person with black hair");

The above query is evaluated as follows:

Create table RESULT as:

SELECT COUNT(o_name)

FROM officer

WHERE officer.picture (CONTAINS, "tall person with black hair");

Let us assume that there are 3 tuples in the table T1, that match the query. Then the aggregate functions COUNT returns 3 in the table RESULT.

b. SUM, MAX, MIN, AVG

The aggregate functions SUM, MAX, MIN and AVG are applied to a set or multiset of numeric values and return the sum, maximum, minimum and average of those values. Let us clarify this with an example:

Query: What is the sum, maximum, minimum and average salary of the officers?

An extended SQL statement for the above query can be written as follows:

SELECT SUM(salary), MAX(salary), MIN(salary), AVG(salary)

FROM officer;

The above query is evaluated as follows:

Create table RESULT as:

SELECT SUM(salary), MAX(salary), MIN(salary), AVG(salary)

FROM officer;

5. Multiple Selections

As we mentioned in Chapter II Section E of this thesis, multiple selections can be represented in disjunctive normal form by using the Boolean operator **and** inside each group, and Boolean operator **or** between groups. With our approach for the design of multiple selections it is possible to have arbitrarily many conditions inside each group and arbitrarily many groups in a query. The design is as follows:

The result of each condition inside a group is retrieved into a temporary table. For each group, these temporary tables are intersected using the set operator INTERSECT; the result of the intersection is put into another temporary table. Later the temporary tables including the results of each group are unioned using the set operator UNION and the result of this operation is put into the table RESULT, which is the final result of the whole query.

Let us elucidate this with an example which includes multiple selections without any nesting condition. An example with nesting condition will be presented in the next subsection.

Query: Retrieve the names, types and pictures of all ships built after 1975 and whose pictures show "nuclear submarine with many missiles" or those whose

displacement is less than 100,000 and whose pictures show "destroyer with many kinds of guided missiles on board".

SQL statement for the above query is as follows:

```
SELECT s_name, type, picture
FROM ship
WHERE (yr_built > 1975 and ship.picture(CONTAINS, "nuclear
    submarine with many missiles")) or (displacement < 100,000
    and ship.picture(CONTAINS, "destroyer with many kinds of
    guided missiles on board"));
```

The query above can be evaluated as follows:

Create table T1 as:

```
SELECT s_name, type, picture
FROM ship
WHERE yr_built > 1975
```

Create table T2 as:

```
SELECT s_name, type, picture
FROM ship
WHERE ship.picture(CONTAINS, "nuclear submarine with many
    missiles");
```

Create table R1 as:

```
T1 INTERSECT T2;
```

Create table T3 as:

SELECT s_name, type, picture

FROM ship

WHERE displacement < 100,000

Create table T4 as:

SELECT s_name, type, picture

FROM ship

WHERE ship.picture(CONTAINS, "destroyer with many kinds

of guided missiles on board"));

Create table R2 as:

T3 INTERSECT T4;

Create table RESULT as:

R1 UNION R2;

6. Complex Queries

So far we presented the design of simple queries, nested queries, set operations and multiple selections. Now we are ready to give an example of a complex query including most of the types of these queries.

Query: List the names and displacements of all ships whose weapons' pictures show "anti_aircraft missile" and whose pictures show "modern air defense cruiser, high speed gas turbine powered ship with many engines" or whose captain's rank is commander and whose executive officers' salary is greater than $35,000.

SQL statement for the above query is:

SELECT s_name, displacement

```
FROM ship
WHERE (s_no IN
    (SELECT s_no
     FROM ship_weapon, weapon
     WHERE ship_weapon.w_name=weapon.w_name
        and weapon.picture(CONTAINS, "anti-aircraft missile"))
        and ship.picture(CONTAINS, "modern air defense cruiser")) or
((EXISTS (SELECT o_id
                FROM officer
                WHERE o_id="capt")
        and (exo_id IN   (SELECT o_id
                FROM officer
                WHERE salary > 35,000))))
```

The above query can be evaluated as follows:

Create table T1 as:

```
SELECT s_no
FROM ship_weapon, weapon
WHERE ship_weapon.w_name=weapon.w_name and weapon.picture
    (CONTAINS, "anti-aircraft missile");
```

Create table T2 as:

```
SELECT o_id
FROM officer
```

WHERE o_id="capt"

Create table T3 as:

SELECT o_id

FROM officer

WHERE salary > 35,000

Create table T4 as:

SELECT s_name, displacement

FROM ship

WHERE s_no IN T1

Create table T5 as:

SELECT s_name, displacement

FROM ship

WHERE ship.picture(CONTAINS, "modern air defense cruiser");

Create table R1 as:

T4 INTERSECT T5

Create table T6 as:

SELECT s_name, displacement

FROM ship

WHERE EXISTS T2

Create table T7 as:

SELECT s_name, displacement

FROM ship

WHERE exo_id IN T3

Create table R2 as:

T6 INTERSECT T7

Create table RESULT as:

R1 UNION R2

Refer to Appendix A of this thesis for a comprehensive example of complex queries.

V. IMPLEMENTATION OF COMPLEX QUERIES

In this chapter, we will first present the user interface for all types of queries supported by the MDBMS prototype, then give the query processing for each type of query in detail and, finally, we will mention the necessary procedures for linking and running the system.

A. USER INTERFACE

In section IV.B., we discussed the design of complex queries using the SQL language. A decision was made to use an interactive interface instead of using an extended version of SQL as the user interface. The idea behind this is to let the casual users use the system more easily. In this section, we present the interface design for the retrieval operations implemented so far by giving examples, rather than describing the user interface in an abstract manner.

1. Simple Queries

According to our classification, a simple query is a query involving one or more conditions in the WHERE clause of an SQL statement with the Boolean operator **and** between conditions. Further, none of these conditions include a nesting condition. Let us clarify what we have just said with an example:

Query: Retrieve the name, rank, salary, picture and voice recording of the commanding officers who reported for duty before 1989 and who are stationed at ships whose pictures show "gas turbine powered ship".

SQL statement for the above query:

```
SELECT  o_name, rank, salary, picture, voice
FROM    officer, ship
WHERE   officer.o_id=ship.exo_id and rep_yr<1989 and
            ship.picture(CONTAINS, "gas turbine powered ship");
```

When the user wants to specify such a query in the MDBMS, he will first select the option 'retrieve' from the main menu. 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 simple query above (the scripts in bold type represent the user's responses).

```
            Multimedia Database Management System
=============================================================
      1. Create Table
      2. Insert Tuple
      3. Retrieve
      4. Delete
      5. Modify
      6. Print out current data information(test purpose)
      0. Quit
=============================================================
      Select your choice :: 3

Your Selection is RETRIEVAL!
Enter table name to hold the temporary result of the query: temp
Select the table(s) separate by comma <,> : (<?> for HELP!)
SELECT TABLE(S): ship, officer

Please enter your join condition
(<?> for help!) : ship.capt_id=officer.o_id

Table ship
Select the attribute(s) separated by comma <,> : (<?> for HELP!)
(Hit <ESC>for no attribute)
SELECT ATTRIBUTE(S) : <ESC>
```

```
Table officer
Select the attribute(s) separated by comma <,> : (<?> for HELP!)
(Hit <ESC> for no attribute)
SELECT ATTRIBUTE(S)  : o_name, rank, salary, photo, voice

Any condition ? (y/n):   y
Group condition ? (y/n): y

        Retrieval Operations Menu
        =========================
        0. Simple Condition
        1. table1 where EXISTS table2
        2. table1 where NOT EXISTS table2
        3. table1 IN table2
        4. table1 NOT IN table2
        =========================
          Select your choice :: 0

Enter table name:  officer
Enter attribute:     rep_yr
Enter the condition: <1989
Where rep_yr <1989

There are 4 records that match the query
record id 1  o_name:Jeff Kulp        rank:Capt  salary:10000  photo id is 1  voice id is 1
record id 2  o_name:Dan Hendricks rank:Cdr   salary:8500   photo id is 2  voice 2
record id 3  o_name:Yavuz Atila     rank:Cdr   salary:7500   photo id is 3  voice 3
record id 4  o_name:John Daley      rank:Cdr   salary:9000   photo id is 4  voice 4

Do you want to see any image data ? (y/n): n
End group ? :n

        Retrieval Operations Menu
        =========================
        0. Simple Condition
        1. table1 where EXISTS table2
        2. table1 where NOT EXISTS table2
        3. table1 IN table2
        4. table1 NOT IN table2
        =========================
        Select your choice :: 0

Your Selection is Simple Condition
Enter table name:  ship
Enter attribute:     picture
Please enter your query description
 * noun phrases separate by commas and end with an exclamation mark
 * sentence end with a period.
(end whole description with an empty line):
gas turbine powered ship!
```

Searching .....
Below is the result of the first 2 conditions in group 1 :
There are 2 records that match the query
record id 1 o_name:Yavuz Atila rank:Cdr salary:7500 photo id is 3 voice 3
record id 2 o_name:John Daley rank:Cdr salary:9000 photo id is 4 voice 4

Do you want to see any image data ? (y/n):n
End group ?:y

Below is the result of group 1 :
record id 1 o_name:Yavuz Atila rank:Cdr salary:7500 photo id is 3 voice 3
record id 2 o_name:John Daley rank:Cdr salary:9000 photo id is 4 voice 4

Do you want to see any image data ? (y/n):n
End condition ?:y

Below is the final result of all groups :
record id 1 o_name:Yavuz Atila rank:Cdr salary:7500 photo id is 3 voice 3
record id 2 o_name:John Daley rank:Cdr salary:9000 photo id is 4 voice 4

Do you want to see any image data ? (y/n):y
Which tuple's image do you want to see? (enter record id) : 1

Record no 1 filename :/tmp_mnt/n/virgo/work/mdbms/MDBMS/91163.173948 Show image ....
The following photo has been found:
Number: 1
Description:
>>black hair,big nose,thin body, tall person with glasses!
<<
Do you want to see the photo?: y

*** The photo is displayed on the screen ***

Do you want to see more image data ? (Y/N): n

Which tuple's sound do you want to hear? (enter record id): 2
Sound management
Record no 2
Play the sound ? (y/n): y

*** Sound is play-backed ***

Do you want to hear more sound data ? (Y/N): n
If you want to intersect / union / minus any two tables:

1. INTERSECT two tables
2. UNION two tables
3. MINUS
0. Quit

Select your choice :: 0

More selections at this level ? (y/n): n
More levels ? (y/n): n

Note that after the tuples that match the query are retrieved, the user is asked which tuple's picture he wants to see and which tuple's voice he wants to hear. This is very convenient. In [REF10], the media data was displayed tuple by tuple without asking the user for his choice which was inconvenient for the user of the MDBMS prototype.

Another difference between our interface design and [REF10] is the addition of the 'Retrieval Operations Menu'. This menu is required to ask the user if his condition is a simple one or a nesting condition.

2. Queries Using Nesting Condition

As we mentioned in IV.2, a nested query is a complete SELECT-FROM-WHERE query within the WHERE clause of a another query which is called the outer query. In this subsection, we present the implementation of nested queries using the nesting operators IN, NOT IN, EXISTS and NOT EXISTS.

a. IN

When the user wants to specify a nested query with the nesting operator IN, he should enter the inner query, put the result in a temporary table and then he should enter the outer query. Let us give an example to clarify this:

Query: Retrieve the names, types and pictures of the ships whose weapon's picture shows "high speed guided torpedo".

An extended SQL statement for the above query:

```
SELECT s_name, type, picture
FROM ship, ship_weapon
WHERE ship.s_no=ship_weapon.s_no
    and w_name IN
    (SELECT w_name
    FROM weapon
    WHERE weapon.picture (CONTAINS, "high speed guided
torpedo"));
```

In order to avoid repetition, we will not give all the steps the user has to follow to complete the above query, instead we will explain them.

The above query consists of the inner query (the SELECT-FROM-WHERE query in the parenthesis) and the outer query. The user will first enter the inner query in the same way as the example of simple queries given in section V.A.1. So far, we assume that we have the result of the inner query in the temporary table "temp1". We can, now, present the rest of the steps that the user has to follow to get the final result of the whole query:

```
More selections at this level? (y/n): n
More levels? (y/n): y
Enter table name to hold the temporary result of the query: result

Select the table(s) separate by comma <,> : (<?> for HELP!)
SELECT TABLE(S): ship, ship_weapon

Please enter your join condition
(<?> for help!) : ship.s_no=ship_weapon.s_no

Table ship
Select the attribute(s) separated by comma <,> : (<?> for HELP!)
SELECT ATTRIBUTE(S)
```

(Hit <ESC> for no attribute)
: s_name, type, picture

Table ship_weapon
Select the attribute(s) separated by comma <,> : (<?> for HELP!)
SELECT ATTRIBUTE(S)
(Hit <ESC> for no attribute)
:<ESC>

Any condition ? (y/n): y
Group condition ? (y/n): n

Retrieval Operations Menu

0. Simple Condition
1. table1 where EXISTS table2
2. table1 where NOT EXISTS table2
3. table1 IN table2
4. table1 NOT IN table2

Select your choice :: 3
Your Selection is table1 IN table2

Enter the temp table name related to IN : temp1
Enter attribute for the appropriate table for condition of IN : w_name

Table ** temp1 **
SELECT ATTRIBUTE (only one attribute!): w_name

There is 1 record that match the query
record id 1 s_name : Michigan yr_built : 1982 picture id is 5

Do you want to see any image data ? (y/n): n
Do you want to see more image data ? (Y/N): n

If you want to intersect / union /minus any two tables:

1. INTERSECT two tables
2. UNION two tables
3. MINUS
0. Quit

Select your choice :: 0
More selections at this level ? (y/n): n
More levels ? (y/n): n

b. NOT IN

Let us present the user interface of a nested query including the comparison operator NOT IN by giving an example query first.

Query: Retrieve the name, rank, pictures and voice recording of the commanding officers who are not stationed at ships whose picture shows "gas turbine powered ship".

An extended SQL statement for the above query using the comparison operator NOT IN can be written as follows:

```
SELECT o_name, rank, photo, voice
FROM officer
WHERE o_id NOT IN
    (SELECT capt_id
    FROM ship
    WHERE ship.picture (CONTAINS, "gas turbine powered  ship");
```

As we did for nested queries using the comparison operator IN, we will not repeat all the steps to be followed by the user for specifying the nested query above either, but just point out the differences in the user interface.

We suppose that we have the result of the inner query in the temporary table "temp1". The rest of the user interface to get the result of the above query is as follows:

```
More selections at this level ? (y/n): n
More levels ? (y/n): y
Enter table name to hold the temporary result of the query: result
```

```
Select the table(s) separate by comma <,> : (<?> for HELP!)
SELECT TABLE(S): officer

Table officer
Select the attribute(s) separated by comma <,> : (<?> for HELP!)
SELECT ATTRIBUTE(S)
(Hit <ESC> for no attribute)
: o_name, rank, photo, voice

Any condition ? (y/n): y
Group condition ? (y/n): n

      Retrieval Operations Menu
      ==========================
      0. Simple Condition
      1. table1 where EXISTS table2
      2. table1 where NOT EXISTS table2
      3. table1 IN table2
      4. table1 NOT IN table2
      ==========================
      Select your choice :: 4
Your Selection is table1 NOT IN table2

Enter the temp table name related to NOT IN : result1
Enter attribute for table officer for condition of NOT IN : o_id

Table ** result1 **
SELECT ATTRIBUTE (only one attribute!) : capt_id

There are 3 records that match the query
record id 1    o_name :  Dan Hendricks     rank :  Cdr          photo id is 2  voice 2
record id 2    o_name :  Fred Pong         rank :  Lt Cdr       photo id is 9  voice 9
record id 3    o_name :  Huseyin Aygun     rank :  Lt Cdr       photo id is 8  voice 8

Do you want to see/hear any media data ? (y/n): n
More selections at this level ? (y/n): n
More levels ? (y/n): n
```

c. EXISTS

The comparison operator EXISTS is usually used in conjunction with a correlated nested query. A correlated nested query is a nested query with a join condition related to the outer query. Considering this as a general rule, we ask the user to enter a

join condition between the inner query and the outer query. Nested queries with the comparison operator EXISTS work as follows:

For each tuple of the outer query, the nested query is evaluated; if at least one tuple exists in the result of the nested query then that tuple of the outer query is retrieved.

Let us give an example for a nested query with the nesting operator EXISTS:

Query: Retrieve the name, type and picture of the ships whose weapon's picture shows "long_range missile against land targets".

The extended SQL statement for the above query:

SELECT s_name, type, picture

FROM ship, ship_weapon

WHERE ship.s_no=ship_weapon.s_no

and w_name EXISTS

(SELECT w_name

FROM weapon

WHERE weapon.picture (CONTAINS, "long_range missile against land targets"));

The user interface portion, after the result of the inner query is put in the temporary table "table1", is as follows:

```
More selections at this level ? (y/n): n
More levels ? (y/n): y
Enter table name to hold the temporary result of the query: result
```

```
Select the table(s) separate by comma <,> : (<?> for HELP!)
SELECT TABLE(S): ship, ship_weapon

Please enter your join condition
(<?> for help!) : ship.s_no=ship_weapon.s_no

Table ship
Select the attribute(s) separated by comma <,> : (<?> for HELP!)
SELECT ATTRIBUTE(S)
(Hit <ESC> for no attribute)
: s_name, type, picture

Table ship_weapon
Select the attribute(s) separated by comma <,> : (<?> for HELP!)
SELECT ATTRIBUTE(S)
(Hit <ESC> for no attribute)
: <ESC>

Any condition ? (y/n): y
Group condition ? (y/n): n

      Retrieval Operations Menu
      =========================
      0. Simple Condition
      1. table1 where EXISTS table2
      2. table1 where NOT EXISTS table2
      3. table1 IN table2
      4. table1 NOT IN table2
      =========================
      Select your choice :: 1
Your Selection is table1 where EXISTS table2

Enter the temp table name related to EXISTS : result1

Please enter your join condition
between the appropriate table and ** temp1 ** :ship_weapon.w_name=weapon.w_name

There are 2 records that match the query
record id 1    s_name :  Kitty Hawk        type :  carrier          picture id is 1
record id 2    s_name :  Mississippi       type :  cruiser          picture id is 2
Do you want to see/hear any media data ? (y/n): n
```

d. NOT EXISTS

The comparison operator NOT EXISTS is also used in conjunction with a correlated nested query. NOT EXISTS works as follows:

For each tuple of the outer query, the nested query is evaluated; if the tuple does not exist in the result of the nested query then that tuple of the outer query is retrieved.

As we did for EXISTS, we again ask the user a join condition between the inner query and the outer query.

Query: Retrieve the name and rank of executive officers who did not attend Gulf War and show their photographs.

The extended SQL statement for the above query:

```
SELECT o_name, rank, photo
FROM officer
WHERE NOT EXISTS
    (SELECT *
    FROM ship, mission
    WHERE ship.m_id=mission.m_id
        and mission.m_name="Gulf War");
```

Suppose that the user has already put the result of the inner query in the temporary table "temp1". The rest of the steps to be followed are given below:

```
More selections at this level ? (y/n): n
More levels ? (y/n): y
Enter table name to hold the temporary result of the query: result

Select the table(s) separate by comma <,> : (<?> for HELP!)
SELECT TABLE(S): officer
Table officer

Select the attribute(s) separated by comma <,> : (<?> for HELP!)
SELECT ATTRIBUTE(S)
(Hit <ESC> for no attribute)
```

: o_name, rank, photo

Any condition ? (y/n): y
Group condition ? (y/n): n

Retrieval Operations Menu

0. Simple Condition
1. table1 where EXISTS table2
2. table1 where NOT EXISTS table2
3. table1 IN table2
4. table1 NOT IN table2

Select your choice :: 2
Your Selection is table1 where NOT EXISTS table2

Enter the temp table name related to NOT EXISTS : temp1

Please enter your join condition
between the appropriate table and ** temp1 ** : officer.o_id=ship.exo_id

There are 2 records that match the query
record id 1 o_name : Huseyin Aygun rank : Lt photo id is 1
record id 2 o_name : Yavuz Atila rank : Lt Cdr photo id is 3

Do you want to see any image data ? (y/n): n

Refer to Appendix A of this thesis for a complex query including nested queries and multiple selections.

3. Set Operations

In this subsection we present the user interface for the set operations INTERSECTION, UNION and MINUS. As we did for nested queries we will only point out the differences in the user interface, instead of repeating all the steps to be followed by the user.

a. UNION

Query: List the names, ranks, pictures and voice recordings of all the officers who worked as an executive officer or as a captain on the ships whose pictures show "nuclear submarine with many kinds of guided torpedoes".

Extended SQL statement for this query:

```
(SELECTo_name, rank, picture, voice
FROM    officer, ship
WHERE officer.o_id=ship.exo_id and  ship.picture(CONTAINS,
            "nuclear submarine with many kinds of guided
            torpedoes"))
UNION
(SELECTo_name, rank, picture, voice
FROM    officer, ship
WHERE officer.o_id=ship.capt_id and ship.picture(CONTAINS,
            "nuclear submarine with many kinds of guided
            torpedoes"));
```

The above query consists of two subqueries with the set operator UNION between them. The user who wants to specify such a query will treat each of the two subqueries as simple queries, put their results in temporary tables and then use the set operations menu to get the final result. Now let us assume we have the result of the first subquery in "temp1" and the result of the second subquery in "temp2". The remaining steps to be followed are as follows:

If you want to intersect / union /minus any two tables:

1. INTERSECT two tables
2. UNION two tables
3. MINUS
0. Quit

Select your choice :: 2
Your selection is UNION

Enter the name of the first temp table: temp1
Enter the name of the second temp table: temp2
Enter a temp table name to hold the result of the query: result

There are 2 records that match the query
record id 1 o_name : Rosemary Stewart rank : Lt photo id is 1 voice id is 5
record id 2 o_name : Yavuz Atila rank : Lt Cdr photo id is 3 voice id is 7
Do you want to see/hear any media data ? (y/n): n

b. INTERSECTION

Query: Retrieve the names, power and pictures of all weapons whose pictures show "high speed close range defense weapon" along with the ones located on board the ship "Elliott".

Extended SQL statement for the above query can be written as follows:

```
(SELECTw_name, power, picture
FROM    weapon
WHERE  weapon.picture(CONTAINS, "high speed close range
           defense weapon"))
INTERSECT
(SELECTw_name, power, picture
FROM    ship, ship_weapon, weapon
WHERE ship.s_no=ship_weapon.s_no and
```

ship_weapon.w_name=weapon.w_name);

As we did for UNION, let us assume that we have the result of the first subquery in "temp1" and the result of the second subquery in "temp2". The remaining steps to be followed are as follows:

```
If you want to intersect / union /minus any two tables:
    ==========
    1. INTERSECT two tables
    2. UNION two tables
    3. MINUS
    0. Quit
    ==========
    Select your choice :: 1
Your selection is INTERSECT
Enter the name of the first temp table: temp1
Enter the name of the second temp table: temp2
Enter a temp table name to hold the result of the query: result

There is 1 record that matches the query
record id 1     w_name :  Trident     power :  100          photo id is 1

Do you want to see any image data ? (y/n): n
```

c. **MINUS**

Query: Retrieve the names, ranks, and pictures of all the officers whose picture show "tall person" but not the ones whose pictures show "blond hair".

Extended SQL statement for this query can be written as follows:

(SELECT o_name, rank, picture

FROM officer

WHERE officer.picture(CONTAINS, "tall person"))

MINUS

(SELECT o_name, rank, picture

FROM officer

WHERE officer.picture(CONTAINS, "blond hair"))

Let us suppose that we have the result of the first subquery in "temp1" and the result of the second subquery in "temp2". The remaining steps to be followed are as follows:

```
If you want to intersect / union /minus any two tables:
    1. INTERSECT two tables
    2. UNION two tables
    3. MINUS
    0. Quit
    Select your choice :: 3
Your selection is MINUS

Enter the name of the first temp table: temp1
Enter the name of the second temp table: temp2
Enter a temp table name to hold the result of the query: result

There are 2 records that match the query
record id 1    o_name :  Rosemary Stewart    rank :  Lt          photo id is 1
record id 2    o_name :  Yavuz Atila         rank :  Lt Cdr      photo id is 3

Do you want to see any image data ? (y/n): n
```

4. Aggregate Functions:

As we mentioned in IV.B.4, aggregate functions are built-in functions in INGRES SQL. These are COUNT, SUM, MAX, MIN and AVG. In this subsection we will present the user interface of the aggregate functions.

a. COUNT

The built-in function COUNT returns the number of tuples resulting from a query. Let us give an example to clarify how COUNT works:

Query: How many executive officers are there in the fleet, whose pictures show "tall person with black hair"?

The extended SQL statement for the above query:

SELECT COUNT(o_name)

FROM officer

WHERE officer.picture (CONTAINS, "tall person with black hair");

When the user wants to specify a query as above he should first select the Retrieve option from the Main Menu and then follow the following steps to get the result:

```
Enter table name to hold the temporary result of the query: temp
Select the table(s) separate by comma <,> : (<?> for HELP!)
SELECT TABLE(S): officer

Table officer
Select the attribute(s) separated by comma <,> : (<?> for HELP!)
(Hit <ESC> for no attribute)
SELECT ATTRIBUTE(S)
: CNT(o_name)

Any condition ? (y/n) :y
Group condition ? (y/n): n

      Retrieval Operations Menu
      =========================
      0. Simple Condition
      1. table1 where EXISTS table2
      2. table1 where NOT EXISTS table2
      3. table1 IN table2
      4. table1 NOT IN table2
      ==========================
        Select your choice :: 0

Enter attribute: picture

Please enter your query description
 * noun phrases separate by commas and end with an exclamation mark
 * sentence end with a period.
(end whole description with an empty line):
```

tall person with black hair!

Searching .....

There are 2 records that match the query
record id 1 COUNT(o_name) = 2
record id 2 COUNT(o_name) = 2

b. SUM, AVG, MAX, MIN

The built-in functions SUM, AVG, MAX and MIN are applied to a set or multiset of numeric values and returns the sum, average, maximum and minimum of those values. Let us illustrate this with an example:

Query: Find the sum, average, maximum and minimum of the salaries of the commanding officers who are stationed at ships whose picture shows "nuclear submarine with many different kinds of torpedoes".

An extended SQL statement for the above query can be written as follows:

```
SELECT SUM(salary), AVG(salary), MAX(salary), MIN(salary)
FROM officer, ship
WHERE ship.capt_id=officer.o_id and ship.picture(CONTAINS, "nuclear
      submarine with many different kinds of torpedoes");
```

When the user wants to specify such a query he should first select the Retrieve option from the Main Menu and then follow the following steps:

Enter table name to hold the temporary result of the query: temp

Select the table(s) separated by comma <,> : (<?> for HELP!)
SELECT TABLE(S): ship, officer

Please enter your join condition

(<?> for help!) : **ship.capt_id=officer.o_id**

Table ship
Select the attribute(s) separated by comma <,> : (<?> for HELP!)
(Hit <ESC> for no attribute)
SELECT ATTRIBUTE(S) : <ESC>

Table officer
Select the attribute(s) separated by comma <,> : (<?> for HELP!)
(Hit <ESC> for no attribute)
SELECT ATTRIBUTE(S)
: **sum(salary), avg(salary), max(salary), min(salary)**

Any condition ? (y/n): **y**
Group condition ? (y/n): **n**

Retrieval Operations Menu

0. Simple Condition
1. table1 where EXISTS table2
2. table1 where NOT EXISTS table2
3. table1 IN table2
4. table1 NOT IN table2

Select your choice :: **0**

Enter table name: **ship**
Enter attribute: **picture**

Please enter your query description
* noun phrases separate by commas and end with an exclamation mark
* sentence end with a period.
(end whole description with an empty line):
nuclear submarine with many different types of torpedoes!

Searching .....

Result of the query:
SUM(salary)=15000 AVG(salary)=7500 MAX(salary)=8000 MIN(salary)=7000

B. QUERY PROCESSING

In Chapter IV, the various cases in which an extended query (i.e., a simple query or a complex query) must be decomposed into multiple SQL queries are illustrated. We also presented that this method of decomposition required the generation of temporary relational tables for further processing.

Complex query operations actually require a compiler action to compile the user input into SQL statements for INGRES. In addition to the catalog tables given in [REF8], other tables are also required to keep the various information for the purpose of complex query operations. In this section we present the data structures used for implementing complex queries.

In order to process a given query, the system needs information on the table name, the attribute names, and the data types of the attributes. The table Selection_Array is created for this purpose as mentioned in [REF10]. Since aggregation is required for many database applications, we decided to add the aggregate_type to the Selection_Array, to hold the type of the built-in aggregate functions in SQL. The built-in aggregate functions in SQL are, as mentioned in IV.B.4, COUNT, SUM, AVG, MAX, MIN and these are used in the SELECT-clauses of queries.

The second structure is the Condition_Array table. This structure holds the conditions for the query and contains the table name, attribute name and the condition for each selection. This is also presented in [REF10] in detail.

The third structure used by [REF10] was Group_Array table which holds the index to the Condition_Array table for each group in the query. We did not use this structure to implement complex queries, since we decompose a given complex query into multiple simple queries, put their results in temporary tables and recompose these results to get the final result.

When the user specifies a nested query, the query is decomposed into multiple simple queries beginning from the innermost query. The information about the table

name, attribute names, the data types of the attributes,etc... are put into the data structures mentioned above. After the result is retrieved in a temporary table entered by the user, the data structures are initialized at the end of the loop controlling the selections at the same level or at the end of the outer loop controlling the selections at different levels.

When the user specifies a complex query consisting of multiple selections, the query is again decomposed into multiple simple queries (i.e., a query with one group and arbitrarily many conditions in this group). Each simple query is evaluated separately and its result is put into a system generated temporary table. Finally the results of these simple queries, depending on the Boolean operators **or** or **and** between the groups, are recomposed using the set operations UNION or INTERSECTION to get the final result. Four arrays are used to hold the temporary table names and to let the user enter a query consisting of arbitrarily many groups and arbitrarily many conditions in each group.

As we mentioned in IV.A.1, INGRES does not support host variables. INGRES considers the MDBMS program as an application program. Information received from the user at run time cannot be passed to INGRES via the embedded C SQL statements. To solve this problem, we had to modify the C code generated by INGRES in the precompilation process, when SQL statements have already been transferred into C code, in such a way that variables can be assigned values at run time. The result is then compiled by the C compiler for execution.

C. HOW TO LINK AND RUN THE SYSTEM

The system is built on the SUN workstation on the server "Virgo" at cs.nps.navy.mil under the account /n/virgo/work/mdbms/MDBMS/db. **db** is an object code module ready for execution. The program itself is called **db.sc**. The program db.sc is first precompiled by INGRES SQL precompiler to produce **db.c**. After we get db.c, we have to compile this program using the C compiler into an object code and link it to the INGRES library, Suntools library, Sunwindows library, Sunpixrects library and other subprograms shown in Figure 3.4. The other files needed in the same directory are prolog_parser, imagei_image_facts and diction.add. To make the link process simpler, a Makefile is used as shown in Figure 5.1.

```
MDBMS_PATH = /n/virgo/work/mdbms/MDBMS
PLPATH = /n/virgo/work/mdbms/MDBMS/PROLOG_SOURCE
OBJMODS=ISfunctions.o ISsubroutine.o rpc_pl_call.o plcall_xdr.o\
        plcall_clnt.o CatalogManagement.o SoundModule.o\
        UserInterface.o CreateModule.o InsertModule.o Retrieve.o\
        ImageModule.o
PLMODS = $(PLPATH)/dict.pl \
        $(PLPATH)/diction.pl \
        $(PLPATH)/interface.pl \
        $(PLPATH)/simple.pl \
        $(PLPATH)/list_util.pl \
        $(PLPATH)/read_capt.pl \
        $(PLPATH)/variable.pl \
        $(PLPATH)/gen_util.pl \
        $(PLPATH)/number.pl \
        $(PLPATH)/semantics.pl
DEFINE =  defines.h errors.h
Global =  GlobalVariables.h
RPC =     plcall.h
FLAGS =   -g
SERVER =  ai9
RSH =     rsh
LINT =    lint
FILES =   Makefile \
        rpc_pl_server.c \
        rpc_pl_call.c \
        plcall.h \
```

```
            plcall_svc.c \
            plcall_xdr.c \
            plcall_clnt.c \
            IMPORTANT_FILES \
            defines.h \
            errors.h \
            ISsubroutine.c \
            ISfunctions.c \
            comcprolog_neu.c \

.c.o:; cc -c $(FLAGS) -o $@ $*.c
Retrieve.o CreateModule.o InsertModule.o CatalogManagement.o \
UserInterface.o SoundModule.o  ImageModule.o: $(Global)

rpc_pl_call.o rpc_pl_server plcall_svc.o plcall_xdr.o plcall_clnt.o: $(RPC)
Retrieve.o CreateModule.o InsertModule.o CatalogManagement.o \
UserInterface.o SoundModule.o ISfunctions.o ISsubroutine.o \
rpc_pl_call.o rpc_pl_server  ImageModule.o: $(DEFINE)

db: db.o $(OBJMODS)
        @echo "creating DATABASE ..."
        cc $(FLAGS) db.o \
        $(OBJMODS) \
        /ingres/lib/libqlib /ingres/lib/compatlib \
        -lsuntool -lsunwindow -lpixrect -lm \
        -o db

db.c: db.sc
        esqlc db.sc

plcall_xdr_sun4.o: plcall_xdr.c
        $(RSH) $(SERVER) cc -c $(FLAGS) \
        -o $(MDBMS_PATH)/plcall_xdr_sun4.o \
        $(MDBMS_PATH)/plcall_xdr.c

plcall_svc_sun4.o: plcall_svc.c
        $(RSH) $(SERVER) cc -c $(FLAGS) \
        -o $(MDBMS_PATH)/plcall_svc_sun4.o \
        $(MDBMS_PATH)/plcall_svc.c

rpc_pl_server: rpc_pl_server.c \
plcall_svc_sun4.o \
plcall_xdr_sun4.o \
comcprolog_neu.c \
$(DEFINE)
        @echo "creating rpc_pl_server ..."
        $(RSH) $(SERVER) cc $(FLAGS) $(MDBMS_PATH)/rpc_pl_server.c \
        $(MDBMS_PATH)/plcall_svc_sun4.o \
        $(MDBMS_PATH)/plcall_xdr_sun4.o \
        -o $(MDBMS_PATH)/rpc_pl_server
```

```
prolog_parser: $(PLMODS) $(PLPATH)/diction.add
        @echo "creating prolog_parser ..."
        sort $(PLPATH)/diction.body $(PLPATH)/diction.add -o $(PLPATH)/diction
        cat $(PLPATH)/diction.head $(PLPATH)/diction > $(PLPATH)/diction.pl
        rm $(PLPATH)/diction.qof
        $(RSH) $(SERVER) qpc -c $(PLPATH)/diction.pl
        $(RSH) $(SERVER) qpc -D $(PLPATH)/interface -o $(PLPATH)/prolog_parser
        mv $(MDBMS_PATH)/prolog_parser $(MDBMS_PATH)/prolog_parser.lastVersion
        cp $(PLPATH)/prolog_parser $(MDBMS_PATH)/prolog_parser

lint:   *.c
        $(LINT) $?
        @touch lint

print:  $(FILES)
        @echo "Print the following files:"
        @ls $?
        @echo "Interrupt with Control c"
        @sleep 3
        pr $? | print
        @touch print
```

Figure 5.1. Makefile

When the user of the MDBMS prototype wants to compile and link a new implementation of db, he must just type "make db" at shell prompt. The execution module will be named db. In the rest of this section we present detailed information about the Unix utility **make**.

Make is a command generator. It generates a sequence of commands for execution by the Unix shell. Make is mostly used to sort out dependency relations among files. For example, a program must be generated linking object files and libraries, which in turn must be created from a programming language source files. If we modify one or more source files, we must re-link the program after recompiling all the sources which are dependent on the modified files. This process is normally repeated many times during the course of a project.

It is this process that "make" greatly simplifies. By recording once and for all the specific relationships among a set of files, we can thereafter let make automatically perform all updating tasks. We need only to issue the command:

$ make db

Make then carries out those tasks necessitated by the project work since the previous make command. It achieves this by examining the file system to determine when the relevant files were last modified. For example, if file A depends on file B, and if file B was modified after file A, then file A must be "re-made"-compiled, linked, or whatever.

We must define the dependencies between modules or files in a **description file**. This file is normally given the name **Makefile**. A description file consists of many entries. Each entry consists of a line containing a colon (the **dependency line**) and one or more **command lines** beginning with a tab. To the left of the colon on the dependency line are one or more **targets**; to the right of the colon are component files on which the targets depend. The tab-indented command lines then show how to make the targets out of their components. For example in Figure 5.1:

db.c: db.sc

esqlc db.sc

means that db.c depends on the file db.sc. db.sc is executed (i.e., the program db.sc is precompiled by the INGRES SQL precompiler) only if db.sc is modified after the last time db.c was made.

We can use any legitimate shell commands and filename pattern_matching characters in a description file. For example some of the shell commands and filename

pattern-matching characters used in the description file in Figure 5.1 are sort, cat, rm, mv, cp and *.c.

In a description file, we can use some **macro definitions**. A macro definition is a line containing an equals sign (=) and not preceded by a colon or a tab. Typically, macro definitions are grouped together at the beginning of the description file. The name to the left of the equals sign is assigned the string of characters following the equals sign. For example the line:

MDBMS_PATH = /n/virgo/work/mdbms/MDBMS

is a macro definition and subsequent references to

$(MDBMS_PATH)

are interpreted as

/n/virgo/work/mdbms/MDBMS

make also defines several "internal macros" that can simplify the description file. One of them is **$?**. **$?** evaluates to the list of components that are younger (i.e., more recently modified) than the current target.

$@ evaluates to the current target name - that is, the target being made.

In a description file, we can define suffix rules, which greatly reduces the complexity of our description files. For example, the suffix rule .c.o in Figure 5.1 describes how to make a .o file from a .c file.

Finally, an important command-line usage of make is:

$make -f Makerose dbrose

which tells make to use the file "Makerose" as a description file to generate the target "dbrose".

VI. CONCLUSION AND SUMMARY

Multimedia database management systems manage multimedia data such as image data and sound data in addition to formatted data. In this thesis, a prototype has been developed maintaining the standard data and media data to implement complex queries.

This thesis outlined some sample applications in which multimedia data is required and presented the design and implementation of complex queries (i.e., nesting conditions and multiple selections) in addition to set operations (UNION, INTERSECTION and MINUS) and aggregate functions (COUNT, SUM, AVG, MAX, MIN).

Having a nested query means a complete SELECT-FROM-WHERE query is within the WHERE-clause of another query. Nested queries are evaluated beginning from the innermost query to the outer queries. The intermediate result at each level is put into a temporary table and then the query at the next level is evaluated until the final result is received.

Multiple selections refer to queries with arbitrarily many groups in the query and arbitrarily many conditions in each group. Each group is evaluated and its result is put into a temporary table. The results of all groups are recomposed using the set operations UNION or INTERSECTION to get the final result.

Besides UNION and INTERSECTION, the set operation MINUS is implemented to let the user evaluate the difference of two tables.

Finally aggregate functions COUNT, SUM, AVG, MAX, MIN are implemented to make it possible to find the number of tuples, the sum, average, maximum and minimum of values for a given query.

An interactive interface was implemented instead of using an extended version of SQL as the user interface. The idea behind this is to let the casual users use the system more easily.

At present only sound data and image data are supported as media data types. However, it is possible to extend the capability of the system to handle other media types such as video, text, signals in a similar manner.

Future works will concentrate on implementation of a graphical user interface for the system, the help utility and the transaction processing. For graphical user interface issue, some research is being done to find the best tool to implement the design done by [REF9]. After the graphical user interface is implemented, it is considered that a help utility for the entire system would let users with little background to use the system more easily.

LIST OF REFERENCES

[REF1] Atila, Y. V., "Design and Implementation of a Multimedia DBMS: Sound Management Integration", Master's Thesis, Naval Postgraduate School, Department of Computer Science, Monterey, Califomia, December 1990.

[REF2] Bertino, E., Gibbs, S., Rabitti, F., Thanos, C. and Tsichritzis, D., "A Multimedia Document Server," in Proc. 6th Japanese Advanced Database Symposium (Tokyo, August 1986), Information Processing Society of Japan, pp.123-134, 1986.

[REF3] Bertino, E., Rabitti, F. and Gibbs, S., "Query Processing in a Multimedia Document System," ACM Trans. on Office Information Systems, v.6,no.1, pp.1-41, January 1988.

[REF4] 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, v.4, no.4, pp. 345-383, October 1986.

[REF5] 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, pp. 57-66, Washington, 1987.

[REF6] Lum, V.Y. and Meyer-Wegener, K., "A Multimedia Database Management System Supporting Content Search in Media Data," Report no. NPS-52-89-020, Naval Postgraduate School, Department of Computer Science, Monterey, California, March 1989.

[REF7] Meyer-Wegener, K., Lum, V.Y. and Wu, C.T., "Managing Multimedia Data - An Exploration," Report no. NPS-52-88-010, Naval Postgraduate School, Department of Computer Science, Monterey, Califomia, March 1988. Also published in Visual Database Systems, Kunii, T.L., pp. 497-523, The North-Holland Publishing Company, Inc., 1989.

[REF8] Pei, S., Design and Implementation of a Multimedia DBMS: Catalog Management, Table Creation and Data Insertion, Master's Thesis, Naval

Postgraduate School, Department of Computer Science, Monterey, California, December 1990.

[REF9] Peabody, C., Design and Implementation of a Multimedia DBMS: Graphical User Interface Design and Implementation, Master's Thesis, Naval Postgraduate School, Department of Computer Science, Monterey, California. (in preparation)

[REF10] Pongsuwan, W., Design and Implementation of a Multimedia DBMS: Retrieval Management, Master's Thesis, Naval Postgraduate School, Department of Computer Science, Monterey, California, September 1990.

[REF11] Rowe, N. and Guglielmo E., "Exploiting Captions for Access to Multimedia Databases", to appear in IEEE Computer, 1991.

[REF12] Stewart, R., Design and Implementation of a Multimedia DBMS: Modification and Deletion, Master's Thesis, Naval Postgraduate School, Department of Computer Science, Monterey, California. (in preparation)

[REF13] Wu, C.T., Nardi, P., Turner, H., Antonopoulos D., "ARGOS Next Generation Shipboard Information Management System," Report no. NPS-52-90-006, Naval Postgraduate School, Department of Computer Science, Monterey, California, December 1989.

[REF14] Woelk, D. and Kim, W., "Multimedia Management in an Object-Oriented Database System," Proc. 13th International Conference on VLDB, Brighton, England, September 1987.

[REF15] Woelk D. et al., "An Object-oriented Approach to Multimedia Databases", Proc. ACM SIGIR Conference, pp. 247-253, 1986.

[REF16] Woelk D. and W. Kim, "Multimedia Information Management in an Object-oriented Database System", Proc. of VLDB, Brighton, England, Sept.1987.

APPENDIX A - A COMPREHENSIVE EXAMPLE OF DESIGN AND USER INTERFACE OF COMPLEX QUERIES

Since it is usually very difficult to express complex queries in words, we will present an artificial example to show how a complex query with a couple of groups and a couple of conditions in each group, some being simple conditions some nesting conditions, is evaluated according to the design we have presented in Chapter IV of this thesis.

SQL Query:

```
SELECT o_name, picture, voice
FROM officer
WHERE ( o_id IN
    (SELECT exo_id
    FROM ship
    WHERE (EXISTS
        (SELECT *
        FROM ship-weapon, ship
        WHERE ( w_name IN
            (SELECT w_name
            FROM weapon
            WHERE fire_range < 100 and weapon.picture (CONTAINS, "high
```

```
                speed guided torpedo"))
        and ship.s_no = ship_weapon.s_no
        and s_no = "SSBN 727"
        and s_no IN
            (SELECT s_no
            FROM ship
            WHERE yr_built > 1975))
        or (NOT EXISTS
            (SELECT *
            FROM ship, ship_weapon
            WHERE ship.s_no=ship_weapon.s_no and displacement > 15,000)
        and w_name IN
            (SELECT w_name
            FROM weapon
            WHERE weapon.picture (CONTAINS, "long_range missile
                against land targets")))
    and ship.picture (CONTAINS, "nuclear submarine with many missiles"))
    or (type = "cruiser")
and salary > 6000)
or ( NOT EXISTS
    (SELECT *
    FROM ship, officer
```

WHERE officer.o_id=ship.exo_id and ship.picture (CONTAINS, "gas turbine powered ship")
and yr_built < 1975));

The above query consists of four nesting levels and is evaluated as follows:

Create Table T1 as:

SELECT w_name
FROM weapon
WHERE fire_range < 100 and weapon.picture (CONTAINS, "high speed guided torpedo");

Create Table T2 as:

SELECT s_no
FROM ship
WHERE yr_built > 1975;

Create Table T3 as:

SELECT *
FROM ship, ship_weapon
WHERE ship_weapon.s_no=ship.s_no and displacement > 15,000;

Create Table T4 as:

SELECT w_name
FROM weapon
WHERE weapon.picture (CONTAINS, "long_range missile against land targets");

Create Table T5 as:

```
SELECT *
FROM ship_weapon, ship
WHERE ship.s_no=ship_weapon.s_no and w_name IN T1;
```

Create Table T6 as:

```
SELECT *
FROM ship_weapon,ship
WHERE ship.s_no=ship_weapon.s_no
    and s_no="SSBN 727";
```

Create Table T7 as:

```
SELECT *
FROM ship_weapon, ship
WHERE ship.s_no=ship_weapon.s_no and s_no IN T2;
```

Create Table T8 as:

```
SELECT *
FROM ship_weapon, ship
WHERE ship_weapon.s_no=ship.s_no and NOT EXISTS T3;
```

Create Table T9 as:

```
SELECT *
FROM ship_weapon, ship
WHERE ship.s_no=ship_weapon.s_no and w_name IN T4;
```

Create Table R1 as:

(T5 INTERSECT T6 INTERSECT T7) UNION (T8 INTERSECT T9);

Create Table T10 as:

SELECT exo_id

FROM ship

WHERE EXISTS R1;

Create Table T11 as:

SELECT exo_id

FROM ship

WHERE ship.picture (CONTAINS, "nuclear submarine with many missiles");

Create Table T12 as:

SELECT exo_id

FROM ship

WHERE type="cruiser";

Create Table R2 as:

(T10 INTERSECT T11) UNION T2

Create Table T13 as:

SELECT *

FROM ship, officer

WHERE officer.o_id=ship.exo_id

and ship.picture (CONTAINS, "gas turbine powered ship")

and yr_built < 1975;

Create Table T14 as:

SELECT o_name, picture, voice

FROM officer

WHERE o_id IN R2;

Create Table T15 as:

SELECT o_name, picture, voice

FROM officer

WHERE salary > 6000);

Create Table T16 as:

SELECT o_name, picture, voice

FROM officer

WHERE NOT EXISTS T13;

Create Table RESULT as:

(T14 INTERSECT T15) UNION T16;

The user interface for the SQL query given at the beginning of this Appendix is as follows:

Multimedia Database Management System

1. Create Table
2. Insert Tuple
3. Retrieve
4. Delete
5. Modify
6. Print out current data information(test purpose)
0. Quit

Select your choice :: 3

Your Selection is RETRIEVAL!
Enter table name to hold the temporary result of the query: T1
Select the table(s) separate by comma <,> : (<?> for HELP!)
SELECT TABLE(S): weapon

Table weapon
Select the attribute(s) separated by comma <,> : (<?> for HELP!)
(Hit <ESC>for no attribute)
SELECT ATTRIBUTE(S) : **w_name**

Any condition ? (y/n): y
Group condition ? (y/n): y

Retrieval Operations Menu

0. Simple Condition
1. table1 where EXISTS table2
2. table1 where NOT EXISTS table2
3. table1 IN table2
4. table1 NOT IN table2

Select your choice :: **0**

Enter attribute: **fire_range**
Enter the condition: **<100**
Where fire_range < 100

There are 3 records that match the query
record id 1 w_name:Vulcan Phalanx
record id 2 w_name:Sea Sparrow
record id 3 w_name:Mk48 Torpedo

End group ? :**n**

Retrieval Operations Menu

0. Simple Condition
1. table1 where EXISTS table2
2. table1 where NOT EXISTS table2
3. table1 IN table2
4. table1 NOT IN table2

```
=========================
Select your choice :: 0

Your Selection is Simple Condition
Enter attribute: picture
Please enter your query description
 * noun phrases separate by commas and end with an exclamation mark
 * sentence end with a period.
(end whole description with an empty line):
high speed guided torpedo!
Searching .....
Below is the result of the first 2  conditions in group 1 :
record id 1  w_name:Mk48 Torpedo

End group ?:y

Below is the result of group 1 :
record id 1  w_name:Mk48 Torpedo

End condition ?:y

Below is the final result of all groups :
record id 1  w_name:Mk48 Torpedo

If you want to intersect / union / minus any two tables:
=======================
1. INTERSECT two tables
2. UNION two tables
3. MINUS
0. Quit
=======================
Select your choice :: 0

More selections at this level ? (y/n): y

Enter table name to hold the temporary result of the query: T2
Select the table(s) separate by comma <,> : (<?> for HELP!)
SELECT TABLE(S): ship

Table ship
Select the attribute(s) separated by comma <,> : (<?> for HELP!)
(Hit <ESC>for no attribute)
SELECT ATTRIBUTE(S)  : s_no
```

Any condition ? (y/n): y
Group condition ? (y/n): n

Retrieval Operations Menu
==========
0. Simple Condition
1. table1 where EXISTS table2
2. table1 where NOT EXISTS table2
3. table1 IN table2
4. table1 NOT IN table2
==========
Select your choice :: **0**

Enter attribute: **yr_built**
Enter the condition: **>1975**
Where yr_built>1975

There are 2 records that match the query
record id 1 s_no:DDG967
record id 2 s_no:SSBN727

More selections at this level ? (y/n): **y**

Enter table name to hold the temporary result of the query: **T3**
Select the table(s) separate by comma <,> : (<?> for HELP!)
SELECT TABLE(S): **ship, ship_weapon**

Please enter your join condition
(<?> for help!) : **ship.s_no=ship_weapon.s_no**

Table ship
Select the attribute(s) separated by comma <,> : (<?> for HELP!)
(Hit <ESC>for no attribute)
SELECT ATTRIBUTE(S) : **s_no**

Table ship_weapon
Select the attribute(s) separated by comma <,> : (<?> for HELP!)
(Hit <ESC>for no attribute)
SELECT ATTRIBUTE(S) : **<ESC>**

Any condition ? (y/n): y
Group condition ? (y/n): n

Retrieval Operations Menu

0. Simple Condition
1. table1 where EXISTS table2
2. table1 where NOT EXISTS table2
3. table1 IN table2
4. table1 NOT IN table2

Select your choice :: **0**

Enter attribute: **displacement**
Enter the condition: **>15,000**

There are 2 records that match the query
record id 1 s_no:CV63
record id 2 s_no:SSBN727

More selections at this level ? (y/n): **y**

Enter table name to hold the temporary result of the query: **T4**
Select the table(s) separate by comma <,> : (<?> for HELP!)
SELECT TABLE(S): **weapon**

Table weapon
Select the attribute(s) separated by comma <,> : (<?> for HELP!)
(Hit <ESC>for no attribute)
SELECT ATTRIBUTE(S) : w_name

Any condition ? (y/n): **y**
Group condition ? (y/n): **n**

Retrieval Operations Menu

0. Simple Condition
1. table1 where EXISTS table2
2. table1 where NOT EXISTS table2
3. table1 IN table2
4. table1 NOT IN table2

Select your choice :: **0**

Enter attribute: **picture**
Please enter your query description

* noun phrases separate by commas and end with an exclamation mark
* sentence end with a period.
(end whole description with an empty line):
long_range missile against land targets!

Searching .....

There is 1 record that match the query
record id 1 w_no:Tomahawk

More selections at this level ? (y/n): **n**
More levels ? (y/n): **y**

Enter table name to hold the temporary result of the query: **R1**

Select the table(s) separate by comma <,> : (<?> for HELP!)
SELECT TABLE(S): **ship_weapon, ship**

Please enter your join condition
(<?> for help!) : **ship.s_no=ship_weapon.s_no**

Table ship_weapon
Select the attribute(s) separated by comma <,> : (<?> for HELP!)
SELECT ATTRIBUTE(S)
(Hit <ESC> for no attribute)
: **s_no**
Table ship
Select the attribute(s) separated by comma <,> : (<?> for HELP!)
SELECT ATTRIBUTE(S)
(Hit <ESC> for no attribute)
: **<ESC>**

Any condition ? (y/n): **y**
Group condition ? (y/n): **y**

Retrieval Operations Menu
==========

0. Simple Condition
1. table1 where EXISTS table2
2. table1 where NOT EXISTS table2
3. table1 IN table2
4. table1 NOT IN table2

```
Select your choice :: 3
Your Selection is table1 IN table2

Enter the temp table name related to IN : T1
Enter attribute for table ship_weapon for condition of IN : w_name

Table ** T1 **
SELECT ATTRIBUTE (only one attribute!): w_name

There is 1 record that match the query
record id 1    s_no : SSBN727

End group ? :n

        Retrieval Operations Menu
        0. Simple Condition
        1. table1 where EXISTS table2
        2. table1 where NOT EXISTS table2
        3. table1 IN table2
        4. table1 NOT IN table2
        Select your choice :: 0

Enter table name:ship_weapon
Enter attribute: s_no
Enter the condition: ="SSBN727"

Below is the result of the first 2  conditions in group 1 :
There is 1 record that match the query
record id 1    s_no : SSBN727

End group ? :n

        Retrieval Operations Menu
        0. Simple Condition
        1. table1 where EXISTS table2
        2. table1 where NOT EXISTS table2
        3. table1 IN table2
```

4. table1 NOT IN table2

Select your choice :: 3
Your Selection is table1 IN table2

Enter the temp table name related to IN : T2
Enter attribute for table ship_weapon for condition of IN : s_no

Table ** T2 **
SELECT ATTRIBUTE (only one attribute!): s_no

Below is the result of the first 3 conditions in group 1 :
record id 1 s_no : DDG967
record id 2 s_no : SSBN727

End group ? :y

Below is the result of group 1 :
record id 1 s_no : DDG967
record id 2 s_no : SSBN727

End condition ? :n

Retrieval Operations Menu

0. Simple Condition
1. table1 where EXISTS table2
2. table1 where NOT EXISTS table2
3. table1 IN table2
4. table1 NOT IN table2

Select your choice :: 2
Your Selection is table1 where NOT EXISTS table2

Enter the temp table name related to NOT EXISTS : T3

Please enter your join condition
between table ship_weapon and ** T3 ** : T3.s_no=ship_weapon.s_no

There are 2 records that match the query
record id 1 s_no:DDG967
record id 2 s_no:SSBN727

End group ? (y/n):n

Retrieval Operations Menu

0. Simple Condition
1. table1 where EXISTS table2
2. table1 where NOT EXISTS table2
3. table1 IN table2
4. table1 NOT IN table2

Select your choice :: 3
Your Selection is table1 IN table2

Enter the temp table name related to IN : T4
Enter attribute for table ship_weapon for condition of IN : w_name

Table ** T4 **
SELECT ATTRIBUTE (only one attribute!): w_name

There is one record that match the query:
record id 1 s_no : SSBN727

Below is the result of first two conditions in group 2:
record id 1 s_no : SSBN727

End group ? :y

Below is the result of group 2:
record id 1 s_no : SSBN727

End condition ? (y/n): y

Below is the final result of all groups :
record id 1 s_no : SSBN727

More selections at this level ? (y/n): n
More levels ? (y/n): y

Enter table name to hold the temporary result of the query: R2

Select the table(s) separate by comma <,> : (<?> for HELP!)
SELECT TABLE(S): ship

Table ship
Select the attribute(s) separated by comma <,> : (<?> for HELP!)
SELECT ATTRIBUTE(S)
(Hit <ESC> for no attribute)
: **exo_id**

Any condition ? (y/n): **y**
Group condition ? (y/n): **y**

Retrieval Operations Menu
==========
0. Simple Condition
1. table1 where EXISTS table2
2. table1 where NOT EXISTS table2
3. table1 IN table2
4. table1 NOT IN table2
==========
Select your choice :: **1**
Your Selection is table1 where EXISTS table2

Enter the temp table name related to EXISTS : **R1**

Please enter your join condition
between table ship_weapon and ** R1 ** : **R1.s_no=ship.s_no**

There are 3 records that match the query
record id 1 exo_id:201
record id 2 exo_id:203
record id 3 exo_id:204

End group ? (y/n):**n**

Retrieval Operations Menu
==========
0. Simple Condition
1. table1 where EXISTS table2
2. table1 where NOT EXISTS table2
3. table1 IN table2
4. table1 NOT IN table2
==========
Select your choice :: **0**

Your Selection is Simple Condition
Enter attribute: **picture**
Please enter your query description
* noun phrases separate by commas and end with an exclamation mark
* sentence end with a period.
(end whole description with an empty line):
nuclear submarine with many missiles!

Searching .....

Below is the result of the first 2 conditions in group 1 :
record id 1 exo_id:204

End group ? (y/n): **y**

Below is the result of group 1 :
record id 1 exo_id:204

End condition ? (y/n): **n**

Retrieval Operations Menu

0. Simple Condition
1. table1 where EXISTS table2
2. table1 where NOT EXISTS table2
3. table1 IN table2
4. table1 NOT IN table2

Select your choice :: **0**

Your Selection is Simple Condition
Enter attribute: **type**
Enter condition : **="cruiser"**

There are 2 records that match the query:
record id 1 exo_id:201
record id 2 exo_id:202

End group ? (y/n): **y**

Below is the result of group 2:
record id 1 exo_id:201
record id 2 exo_id:202

End condition ? (y/n): **y**

Below is the final result of all groups:
record id 1 exo_id:201
record id 2 exo_id:202
record id 3 exo_id:204

More selections at this level ? (y/n): **y**

Enter table name to hold the temporary result of the query: **T13**

Select the table(s) separate by comma <,> : (<?> for HELP!)
SELECT TABLE(S): **ship**

Table ship
Select the attribute(s) separated by comma <,> : (<?> for HELP!)
SELECT ATTRIBUTE(S)
(Hit <ESC> for no attribute)
: exo_id

Any condition ? (y/n): **y**
Group condition ? (y/n): **y**

Retrieval Operations Menu

0. Simple Condition
1. table1 where EXISTS table2
2. table1 where NOT EXISTS table2
3. table1 IN table2
4. table1 NOT IN table2

Select your choice :: **0**

Enter attribute: **yr_built**
Enter the condition: **<1975**

There is 2 record that match the query
record id 1 exo_id:100
record id 2 exo_id:101

End group ? :**n**

Retrieval Operations Menu

```
==========================
0. Simple Condition
1. table1 where EXISTS table2
2. table1 where NOT EXISTS table2
3. table1 IN table2
4. table1 NOT IN table2
==========================
Select your choice :: 0

Your Selection is Simple Condition
Enter attribute: picture
Please enter your query description
 * noun phrases separate by commas and end with an exclamation mark
 * sentence end with a period.
(end whole description with an empty line):
gas turbine powered ship!

Searching .....

There are two records that match the query:
record id 1  exo_id:100
record id 2  exo_id:101

End group ? (y/n): y

Below is the result of the first 2  conditions in group 1 :
record id 1  exo_id:100
record id 2  exo_id:101

End group ? (y/n): y

Below is the result of group 1 :
record id 1  exo_id:100
record id 2  exo_id:101

End condition ? (y/n): y

Below is the final result of all groups:
record id 1  exo_id:100
record id 2  exo_id:101

More selections at this level ? (y/n): n
More levels ? (y/n): y
```

Enter table name to hold the temporary result of the query: **RESULT**

Select the table(s) separate by comma <,> : (<?> for HELP!)
SELECT TABLE(S): **officer**

Table officer
Select the attribute(s) separated by comma <,> : (<?> for HELP!)
SELECT ATTRIBUTE(S)
(Hit <ESC> for no attribute)
: **o_name, picture, voice**

Any condition ? (y/n): **y**
Group condition ? (y/n): **y**

Retrieval Operations Menu

0. Simple Condition
1. table1 where EXISTS table2
2. table1 where NOT EXISTS table2
3. table1 IN table2
4. table1 NOT IN table2

Select your choice :: **3**
Your Selection is table1 IN table2

Enter the temp table name related to IN : **R2**
Enter attribute for table officer for condition of IN : o_id

Table ** R2 **
SELECT ATTRIBUTE (only one attribute!): **exo_id**

There is 3 record that match the query
record id 1 o_name:Pongsuwan picture id is 1 voice id is 1
record id 2 o_name:R. Stewart picture id is 2 voice id is 2
record id 3 o_name:H. Aygun picture id is 3 voice id is 3

End group ? :**n**

```
        Retrieval Operations Menu
        =========================
        0. Simple Condition
        1. table1 where EXISTS table2
        2. table1 where NOT EXISTS table2
        3. table1 IN table2
        4. table1 NOT IN table2
        =========================
          Select your choice :: 0

Enter attribute: salary
Enter the condition: >6000

There is 1 record that match the query:
record id 1   o_name:Pongsuwan  picture id is 1  voice id is 1

Below is the result of the first two conditions in group 1:
record id 1   o_name:Pongsuwan  picture id is 1  voice id is 1

End group ? :y

Below is the result of group 1:
record id 1   o_name:Pongsuwan  picture id is 1  voice id is 1

End condition ? (y/n): n

        Retrieval Operations Menu
        =========================
        0. Simple Condition
        1. table1 where EXISTS table2
        2. table1 where NOT EXISTS table2
        3. table1 IN table2
        4. table1 NOT IN table2
        =========================
        Select your choice :: 2
Your Selection is table1 where NOT EXISTS table2

Enter the temp table name related to NOT EXISTS : T13

Please enter your join condition
between table officer and ** T13 ** : officer.o_id=T13.exo_id

There are 5 records that match the query
```

record id 1 o_name:Y. Atila picture id is 8 voice id is 8
record id 2 o_name:R. Stewart picture id is 2 voice id is 2
record id 3 o_name:H. Aygun picture id is 3 voice id is 3
record id 4 o_name:S. Pei picture id is 7 voice id is 7
record id 5 o_name:A. Kara picture id is 9 voice id is 9

End group ? (y/n): y

Below is the result of group 2:
record id 1 o_name:Y. Atila picture id is 8 voice id is 8
record id 2 o_name:R. Stewart picture id is 2 voice id is 2
record id 3 o_name:H. Aygun picture id is 3 voice id is 3
record id 4 o_name:S. Pei picture id is 7 voice id is 7
record id 5 o_name:A. Kara picture id is 9 voice id is 9

End condition ? (y/n): y

Below is the final result of all groups:
record id 1 o_name:Pongsuwan picture id is 1 voice id is 1
record id 2 o_name:R. Stewart picture id is 2 voice id is 2
record id 3 o_name:H. Aygun picture id is 3 voice id is 3
record id 4 o_name:S. Pei picture id is 7 voice id is 7
record id 5 o_name:Y. Atila picture id is 8 voice id is 8
record id 6 o_name:A. Kara picture id is 9 voice id is 9

Do you want to see any image data ? (y/n):y
Which tuple's image do you want to see? (enter record id) : 5

Record no 5 filename :/tmp_mnt/n/virgo/work/mdbms/MDBMS/91163.173948
Show image ....
The following photo has been found:
Number: 5
Description:
>>black hair,big nose,thin body, tall person with glasses!
<<
Do you want to see the photo?: y

*** The photo is displayed on the screen ***

Do you want to see more image data ? (Y/N): n

Which tuple's sound do you want to hear? (enter record id): 2
Sound management

Record no 2
Play the sound ? (y/n): y

*** Sound is play-backed ***

Do you want to hear more sound data ? (Y/N): n

APPENDIX B - PROGRAM CODE OF RETRIEVAL OPERATIONS

```
/*************************************************************************
Title       : Retrieve.c
Author      : Aygun/ Stewart
Date        : August 1991
History     : Improvements on Retrieval operations to include complex
              query processing. Also contains the procedures for modify
              and deletion operations. An if clause provides the ability
              to switch options for retrieval,modify or deletion of data
Description:This module implements the retrieval process in the
              Multimedia Database System.
*************************************************************************
Export Interface :
      retrieve(RTRVE_MODE):
                incorporates the retrieval process. The user is asked to
                enter the name of table(s) and attribute(s) he wants to
                retrieve. If he does not know the names of the tables or
                attributes, he can type "?" to list all the tables and
                attributes in the catalog.
      retrieve(DEL_MODE):
                incorporates the deletion process. The user is asked to
                enter table name and condition for deletion.
      retrieve(MOD_MODE):
                incorporates the modification process. The user is asked to
                enter table name and condition for modification.
*************************************************************************
Import Interface :
      print_all_table() : Prints out the table catalog information on screen
                          from InsertModule.c

      check_table_name(): Checks the table_name if it is duplicate
      get_media_name()  : Get media table name by appending table_key at the
                          end of att_name.
                          from CreateModule.c

      yes_no_answer()   : Gets yes or no answer from the user.
      clr_scr()         : Clears the screen.
                          from UserInterface.c
```

```
        play_sound(filename):Sends command from SUN to PC to play the SOUND    *
                             media file.
                             from SoundModule.c
 ************************************************************************/
#include <stdio.h>
#include <string.h>
#include <pixrect/pixrect_hs.h>
#include <sys/wait.h>
#include <suntool/sunview.h>
#include <suntool/canvas.h>
#include "defines.h"
#include "errors.h"
#include "struct.h"
#include "GlobalVariables.h"
#include <rpc/rpc.h>
#include "plcall.h"
#include "defines.h"
#include "errors.h"

char c;
char temp_media_name[3];
char join_condition[100];
int look_more=0;                /* use for loop the cursor */
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 *all_condition;
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();
char temp_table[20];
char temp_table1[20];
char temp_table2[20];
char temp_table3[5]={'h','u','s','i','4'};
char temp_table4[5]={'h','u','s','i','5'};
char temp_table8[5]={'h','u','s','o','1'};
char temp_table9[5]={'h','u','s','o','2'};
char temp_table10[5]={'h','u','s','o','3'};
char temp_table11[5]={'h','u','s','o','4'};
char group1[3]={'g','r','1'};
char group2[3]={'g','r','2'};
char condition_for_nested[100];
char attribute_for_nested[20];
char join_for_nested[99];
int more_selections;
int more_levels;
int aggregate_found;
char t1[5]={'t','_','a','_','1'};
char t2[5]={'t','_','a','_','2'};
char t3[5]={'t','_','a','_','3'};
char t4[5]={'t','_','a','_','4'};
char wrong_descrp = TRUE;
int act_media_count;
int act_media_list[10];
int media_counter=0;
int formatted_flag;
int image_flag;
int sound_flag;
/*************************************************************************
Procedure initialize the array to empty
Initialize all parameters used in the retrieve to null
*************************************************************************/
void init()
{
      int i,j;
      icond=0;
      gcond=0;
      numgroup=0;
      numcon=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 to process the sound condition
put the result in the media tale [number condition] for process later
/*****************************************************************************
void process_icon3(query_phrase,number)
char query_phrase[DESCRLEN+1];
int number;
{
  int id;
  char answer, repeat, yes_no_answer (),con_number,medianum;
  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 ");
    temp_media_name[0]='p';
    medianum=number+48;
    temp_media_name[1]=medianum;
    temp_media_name[2]=0;
    printf("\n%s",temp_media_name);
    IIwritedb(temp_media_name);
    IIwritedb("(");
    IIwritedb("s_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 ("Searchqing .....\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[ERRLEN+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;
          /*  printf("main.sc(ISfnc1):  %s\n", ISfnc1); */
          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(".s_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)
            {
                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 ");
   IIwritedb(temp_media_name);
   IIwritedb("(s_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: */
                                s q l c a . s q l c o d e     =
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 () */
/*************************************************************************/
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,medianum;
  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 ");
    temp_media_name[0]='p';
    medianum=number+48;
    temp_media_name[1]=medianum;
    temp_media_name[2]=0;
    printf("\n%s",temp_media_name);
    IIwritedb(temp_media_name);
    IIwritedb("(");
    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[ERRLEN+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;
```

```
            /*  printf("main.sc(ISfnc1):  %s\n", ISfnc1); */
            if ( sqlca.sqlcode == NOT_FOUND )
              {
               printf("main.sc: ISshows_fetch liefert 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 ");
    IIwritedb(temp_media_name);
    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: */
                                  s q l c a . s q l c o d e     =
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 () */
/*************************************************************************/
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,temp_table, image_id)
int imageno;
int tupleno;
char temp_table[20];
int image_id;
{
  int desired_tupleno;
  char image_value[20];
  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  IServor;
  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;
  desired_tupleno=tupleno;
  /* 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:
   */
c=1;
inttostr(image_id, image_value);
{
if (IIcsrOpen((char *)0,"cursor_output1","db",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(image_value);
    IIcsrQuery ((char *)0);
  }


while (look_more==0) {
  if (IIcsrFetch((char *)0, "cursor_output1","db") != 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 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);
          IIcsrClose((char *)0,"cursor_output1","db");
       }
     }
     fclose(img_file);
   }
   IIcsrEFetch((char *)0);
   j++;
   if (j==c) {
     look_more = 1;
   };
}
/*IIcsrClose((char *)0,"cursor_output1","db");*/
}
}
/*************************************************************************/
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,temp_table, sound_id)
int soundno;
int tupleno;
char temp_table[20];
int sound_id;
{
  char sound_value[20];
  int desired_tupleno;
  char Answer,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  IServor;
  STR_path file_name;
  char ISfn1[FILENAMELEN+1];
  char ISdescr1[DESCRLEN+1];
  desired_tupleno=tupleno;
  c=1;

  inttostr(sound_id, sound_value);
if (IIcsrOpen((char *)0,"cursor_output1","db4",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(sound_value);
    IIcsrQuery ((char *)0);
  } /* IIcsropen */
  while (look_more==0) {
    if (IIcsrFetch((char *)0, "cursor_output1","db4") != 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);
       IIcsrClose((char *)0,"cursor_output1","db4");
      }
      IIcsrEFetch((char *)0);
      j++;
      if (j==c) {
```

```
      look_more = 1;
     }
    } /* IICSRFECCH */
  } /* end while */

}  /* end of display_sound () */
/*************************************************************************/
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\n\
 * noun phrases separate by commas and end with an exclamation mark\n\
 * sentence end with a period.\n\
(end whole description with an 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);
       } while (query_err);
    strcpy(con[numcon],query_phrase);
    if (contype[numcon]==1) {
    process_icon2 (query_phrase,numcon);
       }
    if (contype[numcon]==2) {
    process_icon3 (query_phrase,numcon);
       }

}
/*****************************************************************
This procedure handles if there are  more than one conditions in the query.
*****************************************************************/
nested_gcondition(choice,temp_table1,temp_table2,temp_table)
char choice;
char temp_table1[20];
char temp_table2[20];
char temp_table[20];
{
  int group_number=0;
  int nested_counter=0;
  int k,l;
       int endgroup,i,more,found=FALSE;
       char ans,ans2;
  endgroup = 0;
  more = 0;
```

```
numcon=0;
numgroup=0;

choice=utility_menu(choice,temp_table1,temp_table2,temp_table);

if (choice=='0'){
  cond=1;
  gcond=0;
}
    while (more != 1)  {
    while (endgroup != 1) {
    for (i=0;i < att_index;i++)
    {
      if (choice=='0'){
        if (m > 1 ) {
          printf("\nEnter table name ");
          gets(tab[numcon]);
          strcpy (table_array[table_index].table_name, tab[numcon]);
        }
      }
          if (m==1) {
            strcpy (tab[numcon], stab[0].t_name);
          }
          if (choice=='0'){
            cond=1;
            gcond=0;
            printf("\nEnter attribute ");
            gets(att[numcon]);
            getatttype(tab[numcon], att[numcon],atttype[numcon]);
            if (strcmp(atttype[numcon],"image")==0)
              {
                contype[numcon]=1;
                process_icon();
              }
            else if (strcmp(atttype[numcon],"sound")==0)
              {
                contype[numcon]=2;
                process_icon();
              }
            else {
              printf("Enter the condition \n");
              gets(con[numcon]);
              contype[numcon]=0;
```

```
    }
  }/*end if choice==0 */

   nested_counter=nested_counter+1;
   if ((nested_counter%2)==1){
     if (choice=='0'){
       cond=1;
       gcond=0;
       ql_retrieve(temp_table8);
/*     ql_printdata(temp_table8);*/
       cond=0;
       numcon=0;
       numgroup=0;
       init_buffer(tab,10);
       init_buffer(att,10);
       for (k=0; k<10; k++){
       for (l=0; l<100; l++){
         con[k][l]='0';
       }
       }
     }
     if (choice=='1'){
       temp1_exists_temp2(temp_table1, temp_table2, temp_table8);
       ql_printdata(temp_table8);
       init_buffer(join_for_nested,99);
     }
     if (choice=='2'){
       temp1_not_exists_temp2(temp_table1, temp_table2, temp_table8);
       ql_printdata(temp_table8);
       init_buffer(join_for_nested,99);
     }
     if (choice=='3'){
       temp1_in_temp2(temp_table1, temp_table2, temp_table8);
       ql_printdata(temp_table8);
       init_buffer(condition_for_nested,100);
       init_buffer(attribute_for_nested,20);
     }
     if (choice=='4'){
       temp1_not_in_temp2(temp_table1, temp_table2, temp_table8);
       ql_printdata(temp_table8);
       init_buffer(condition_for_nested,100);
       init_buffer(attribute_for_nested,20);
```

```
    }
  }/* end if nested_counter%2==1 */


  if ((nested_counter%2)==0){
    if (choice=='0'){
      cond=1;
      gcond=0;
      ql_retrieve(temp_table9);
/*    ql_printdata(temp_table9);*/
      cond=0;
      numcon=0;
      numgroup=0;
      init_buffer(tab,10);
      init_buffer(att,10);
      for (k=0; k<10; k++){
      for (l=0; l<100; l++){
        con[k][l]='0';
      }
      }
    }
    if (choice=='1'){
      temp1_exists_temp2(temp_table1, temp_table2, temp_table9);
      ql_printdata(temp_table9);
      init_buffer(join_for_nested,99);
    }
    if (choice=='2'){
      temp1_not_exists_temp2(temp_table1, temp_table2, temp_table9);
      ql_printdata(temp_table9);
      init_buffer(join_for_nested,99);
    }
    if (choice=='3'){
      temp1_in_temp2(temp_table1, temp_table2, temp_table9);
      ql_printdata(temp_table9);
      init_buffer(condition_for_nested,100);
      init_buffer(attribute_for_nested,20);
    }
    if (choice=='4'){
      temp1_not_in_temp2(temp_table1, temp_table2, temp_table9);
      ql_printdata(temp_table9);
      init_buffer(condition_for_nested,100);
      init_buffer(attribute_for_nested,20);
    }
```

```
            }/* end if nested_counter%2==0 */



          if (nested_counter==2){
/*              printf("\nBelow is the result of the first %d  conditions in group %d :",
nested_counter, group_number+1);*/
/*           printf("\nBefore intersection...nested_counter->%d",nested_counter);*/
            intersect_tables(temp_table8,temp_table9,temp_table10);
/*          ql_printdata(temp_table10);*/
            drop_table(temp_table8);
            drop_table(temp_table9);
          }

          if (nested_counter>2){
            if ((nested_counter%2)==1){
/*              printf("\nBelow is the result of the first %d  conditions in group %d :",
nested_counter, group_number+1);*/
/*           printf("\nBefore intersection...nested_counter->%d",nested_counter);*/
              intersect_tables(temp_table10,temp_table8,temp_table11);
/*            ql_printdata(temp_table11);*/
              drop_table(temp_table8);
              drop_table(temp_table10);
            }

            if ((nested_counter%2)==0){
/*              printf("\nBelow is the result of the first %d  conditions in group %d :",
nested_counter, group_number+1);*/
/*           printf("\nBefore intersection...nested_counter->%d",nested_counter);*/
              intersect_tables(temp_table11,temp_table9,temp_table10);
/*            ql_printdata(temp_table10);*/
              drop_table(temp_table11);
              drop_table(temp_table9);
            }
          }/* end if nested_counter>2 */

    /*    }*/ /* end if '1'<=choice=<'4' -----deneme-----*/

          printf("\nEnd group ?");
          ans=yes_no_answer();
          if ((ans==121)||(ans==89)) {
            group_number=group_number+1;
            if (group_number==1){
```

```
                if (nested_counter==1){
                  union_tables_for_nested(temp_table8, group1);
/*                printf("\nBelow is the result of group %d :",group_number);
                  ql_printdata(group1);*/
                  drop_table(temp_table8);
                 }
                 if (nested_counter>1){
                  if ((nested_counter%2)==0){
                  union_tables_for_nested(temp_table10, group1);
/*                printf("\nBelow is the result of group %d :",group_number);
                  ql_printdata(group1);*/
                  drop_table(temp_table10);
                  }
                  if ((nested_counter%2)==1){
                  union_tables_for_nested(temp_table11, group1);
/*                printf("\nBelow is the result of group %d :",group_number);
                  ql_printdata(group1);*/
                  drop_table(temp_table11);
                  }
                 }/* end if nested_counter > 1 */
                }/*end if group_number==1 */

                if (group_number==2){
                 if (nested_counter==1){
                  union_tables_for_nested(temp_table8, group2);
/*                printf("\nBelow is the result of group %d :",group_number);
                  ql_printdata(group2);*/
                  drop_table(temp_table8);
                 }
                 if (nested_counter>1){
                  if ((nested_counter%2)==0){
                  union_tables_for_nested(temp_table10, group2);
/*                printf("\nBelow is the result of group %d :",group_number);
                  ql_printdata(group2);*/
                  drop_table(temp_table10);
                  }
                  if ((nested_counter%2)==1){
                  union_tables_for_nested(temp_table11, group2);
/*                printf("\nBelow is the result of group %d :",group_number);
                  ql_printdata(group2);*/
                  drop_table(temp_table11);
                  }
                 }/* end if nested_counter>1 */
```

```
                union_tables(group1, group2);
/*              printf("\nBelow is the result of the first %d groups ",group_number);
                ql_printdata(group2);*/
                drop_table(group1);
               }/*end if group_number==2 */

               if (group_number>2){
                 if (nested_counter==1){
                   union_tables_for_nested(temp_table8, group1);
/*                 printf("\nBelow is the result of group %d :",group_number);
                   ql_printdata(group1);*/
                   drop_table(temp_table8);
                 }
                 if (nested_counter>1){
                   if ((nested_counter%2)==0){
                   union_tables_for_nested(temp_table10, group1);
/*                 printf("\nBelow is the result of group %d :",group_number);
                   ql_printdata(group1);*/
                   drop_table(temp_table10);
                   }
                   if ((nested_counter%2)==1){
                   union_tables_for_nested(temp_table11, group1);
/*                 printf("\nBelow is the result of group %d :",group_number);
                   ql_printdata(group1);*/
                   drop_table(temp_table11);
                   }
                 }/* end if nested_counter>1 */
                 union_tables(group1,group2);
/*               printf("\nBelow is the result of the first %d groups ",group_number);
                 ql_printdata(group2);*/
                 drop_table(group1);
               }/* end if group_number > 2 */
               nested_counter=0;
               endgroup=1;
              /* printf("\nGroup      %d",numgroup);
               printf("\nCondition  %d",numcon);*/
               i=600;
              }/* end if ans= YES to end group ? */
              if ((ans==110)||(ans==78)) {
                choice=utility_menu(choice,temp_table1,temp_table2,temp_table);
              }
             } /* End for */
           } /* END WHILE */
```

```
        printf("\nEnd condition ?");
        ans=yes_no_answer();
        if ((ans==121)||(ans==89))
          {
           if (group_number==1){
             union_tables_for_nested(group1, temp_table);
             drop_table(group1);
             printf("\nBelow is the final result :");
             ql_printdata(temp_table);
           }
           if (group_number>1){
             union_tables_for_nested(group2, temp_table);
             drop_table(group2);
             printf("\nBelow is the final result :");
             ql_printdata(temp_table);
           }
/*         if (choice=='0')
             group_count[numgroup].endgroup = numcon-1;*/
           endgroup=1;
           more = 1;
           i=0;
          }/* if ans=YES to end condition? */
        else {
          more=0;
          endgroup=0;
          i=0;
          nested_counter=0;
          choice=utility_menu(choice,temp_table1,temp_table2,temp_table);
        /*    group_count[numgroup].endgroup = numcon-1;
           numgroup=numgroup+1;
           group_count[numgroup].begingroup=numcon;*/

        }/*end else*/

      } /* End more */
    group_number=0;
}
/*************************************************************************
This function handles if there is only one condition in the query.
*************************************************************************/
nested_processcondition(choice,temp_table1,temp_table2,temp_table)
char choice;
char temp_table1[20];
```

```
char temp_table2[20];
char temp_table[20];
{
      char    ans2,a;
      int i,j;
      gcond=0;
      printf("\nGroup condition ? (y/n)  ");
      ans2=yes_no_answer();
      if ((ans2==121)||(ans2==89))
      {
        nested_gcondition(choice,temp_table1,temp_table2,temp_table);
      }
      else
      {

            gcond=0;
            choice=utility_menu(choice,temp_table1,temp_table2,temp_table);
            if (choice == '0'){
              cond=1;
              if (m > 1 ) {
                printf("\nEnter table name ");
                gets(tab[0]);
              }
              if (m==1) {
                strcpy (tab[0], stab[0].t_name);
              }
              printf("\nEnter attribute name ");
              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)
                {
                  contype[0]=1;
                  process_icon();
                }
              else if (strcmp(atttype[0],"sound")==0)
                {
                  contype[0]=2;
                  process_icon();
                }
              else {
                printf("Enter the condition \n");
                gets(con[0]);
```

```
            contype[0]=0;
          }
        }
        else
          cond=0;

        if (choice=='0')
          ql_retrieve(temp_table);
        if (choice=='1')
          temp1_exists_temp2(temp_table1, temp_table2, temp_table);
        if (choice=='2')
          temp1_not_exists_temp2(temp_table1, temp_table2, temp_table);
        if (choice=='3')
          temp1_in_temp2(temp_table1, temp_table2, temp_table);
        if (choice=='4')
          temp1_not_in_temp2(temp_table1, temp_table2, temp_table);
        ql_printdata(temp_table);
        }
}

/*************************************************************************/
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++) {
       if (strcmp(table_array[i].table_name,tab_name)==0) {
         x = i;
         y = table_array[i].att_entry;
         printf("\nTable Name: %s\n",table_array[i].table_name); /* print table name */
           printf("\n**Attribute****Data Type**");
             while (y != -1) {
                  printf("\n%13s    %s",att_array[y].att_name,att_array[y].data_type);
               y = att_array[y].next_index;
             }  /* End while y!=-1 */
             if (y==-1) {
                    printf("\n");
                 i=500;
             } /* Exit loop */
         }   /* End if */
    }  /* End for */
```

```
}
/*******************************************************/
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
/*******************************************************/
ql_retrieve(temp_table)
char temp_table[20];
{
  int d,e,r;
    int i,j,k,l;
    char gnum,medianum,operator[4];
    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 ...);
       */
    k=0;
    i=0;
    j=0;
    l=0;
  r=0;

    IIsqInit((char *)0);
    IIwritedb("retrieve into ");
    IIwritedb(temp_table);
    IIwritedb("(");
    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);
                IIwritedb(")");
            }
        }
       if (cond==1) {
        IIwritedb("where(");
           if (m>1) {
                IIwritedb("(");
                IIwritedb(join_condition);
                IIwritedb(")");
                IIwritedb(" and ");
                }
    if (gcond==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("=");
       temp_media_name[0]='p';
       medianum=0+48;
       temp_media_name[1]=medianum;
       temp_media_name[2]=0;
       IIwritedb(temp_media_name);
                IIwritedb(".");
                IIwritedb("i_id");
                      }
           if (contype[0]==2) {
                IIwritedb(tab[0]);
                IIwritedb(".");
                IIwritedb(att[0]);
                IIwritedb("=");
       temp_media_name[0]='p';
       medianum=0+48;
       temp_media_name[1]=medianum;
       temp_media_name[2]=0;
       IIwritedb(temp_media_name);
                IIwritedb(".");
```

```
            IIwritedb("s_id");
            }
        } /* end if no group */
        IIwritedb(")");
    } /* end if con=1 */
}
/*************************************************************************
This function takes two temp tables and unions them and returns
the result to the calling function
*************************************************************************/
union_tables(temp_table1, temp_table)
char temp_table1[20];
char temp_table[20];
{

  int c=0,j=0,k=0,l=0,temp, count;
  /*char*/ STR_name char_value[21];
  char file_name[20],a;
  int  integer_value,media_value,found,media1_value;
  float real_value;
   int i=0,select=0;
int g=0;
/*printf("\nNow we are in union_tables");*/
/* # line 3169 "db.sc" */  /* select */
  {
    IIsqInit((char *)0);
    IIwritedb("retrieve(c=(count(");
    IIwritedb(temp_table1);
    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 in temp_table %s",c, temp_table1);*/

/* # line 3171 "db.sc" */  /* host code */
```

```
    if (IIcsrOpen((char *)0,"cursor_output","db1",0,temp_table1) != 0) {
      IIwritedb("retrieve(");
      for (select=0;select<n-1;select++) {
      IIwritedb(satt[select].a_name);
      IIwritedb("=");
      IIwritedb(temp_table1);
      IIwritedb(".");
      IIwritedb(satt[select].a_name);
      IIwritedb(",");
       }
      IIwritedb(satt[select].a_name);
      IIwritedb("=");
      IIwritedb(temp_table1);
      IIwritedb(".");
      IIwritedb(satt[select].a_name);
      IIwritedb(")");
      IIcsrQuery((char *)0);
    } /* IIcsrOpen */

/* # line 3169 "db.sc" */  /* select */
  {
    IIsqInit((char *)0);
    IIwritedb("retrieve(g=(count(");
    IIwritedb(temp_table);
    IIwritedb(".");
    IIwritedb(satt[0].a_name);
    IIwritedb(")))");
    IIsqRinit((char *)0);
    if (IIerrtest() == 0) {
      if (IInextget() != 0) {
        IIretdom(1,30,4,&g);
      }
      IIsqFlush((char *)0);
    }
  }

/*printf("\nThere are %d records in temp_table %s",g, temp_table);*/

/* # line 3171 "db.sc" */  /* host code */
    if (IIcsrOpen((char *)0,"cursor_output","db2",0,temp_table) != 0) {
      IIwritedb("retrieve(");
      for (select=0;select<n-1;select++) {
      IIwritedb(satt[select].a_name);
```

```
      IIwritedb("=");
      IIwritedb(temp_table);
      IIwritedb(".");
      IIwritedb(satt[select].a_name);
      IIwritedb(",");
       }
      IIwritedb(satt[select].a_name);
      IIwritedb("=");
      IIwritedb(temp_table);
      IIwritedb(".");
      IIwritedb(satt[select].a_name);
      IIwritedb(")");
      IIcsrQuery((char *)0);
    }

/*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 */

/* # line 7 "insert.sc" */  /* insert */
  {
    while (look_more == 0) {
      if (IIcsrFetch((char *)0,"cursor_output","db1") != 0) {
       IIsqInit((char *)0);
       IIwritedb("append to ");
       IIwritedb(temp_table);
       IIwritedb("(");
/*     printf("\nrecord id %d \t",l+1);*/
       for (i=0;i<n-1;i++) {
         IIwritedb(satt[i].a_name);
         IIwritedb("=");
         if (strcmp(satt[i].data_type,"c20")==0) {
           IIcsrRet(1,32,0, char_value);
/*         printf("%s :  %s",satt[i].a_name,char_value);*/
           IIsetdom(1,32,0, 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);*/
          IIsetdom(1,30,4,&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);*/
          IIsetdom(1,31,4,&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);*/
          IIsetdom(1,30,4,&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);*/
          IIsetdom(1,30,4,&media1_value);
        }
        IIwritedb(",");
      }
      IIwritedb(satt[i].a_name);
      IIwritedb("=");
      if (strcmp(satt[i].data_type,"c20")==0) {
        IIcsrRet(1,32,0, char_value);
/*      printf("%s :  %s",satt[i].a_name,char_value);*/
        IIsetdom(1,32,0, 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);*/
        IIsetdom(1,30,4,&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);*/
        IIsetdom(1,31,4,&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);*/
        IIsetdom(1,30,4,&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);*/
          IIsetdom(1,30,4,&media1_value);
        }

/*      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 */
        }
        IIwritedb(" )");
        IIsqSync(3,(char *)0);
       } /* IIcsrFetch */
     } /* end while */
   }
   IIcsrClose((char *)0,"cursor_output","db1"); /* close the cursor */
   IIcsrClose((char *)0,"cursor_output","db2"); /* close the cursor */
   return(temp_table);
}
/*************************************************************************
This function takes two temp tables and unions them, puts the result in
temp_table1 and returns the result to the calling function
*************************************************************************/
union_tables_for_nested(temp_table1, temp_table)
char temp_table1[20];
char temp_table[20];
{

  int c=0,j=0,k=0,l=0,temp, count;
  /*char*/ STR_name char_value[21];
  char file_name[20],a;
  int  integer_value,media_value,found,media1_value;
  float real_value;
   int i=0,select=0;
int g=0;
/*printf("\nNow we are in union_tables_for_nested");*/
/* # line 3169 "db.sc" */  /* select */
  {
    IIsqInit((char *)0);
    IIwritedb("retrieve(c=(count(");
```

```
    IIwritedb(temp_table1);
    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 in temp_table %s",c, temp_table1);*/

/* # line 3171 "db.sc" */  /* host code */
    if (IIcsrOpen((char *)0,"cursor_output","db1",0,temp_table1) != 0) {
      IIwritedb("retrieve(");
      for (select=0;select<n-1;select++) {
      IIwritedb(satt[select].a_name);
      IIwritedb("=");
      IIwritedb(temp_table1);
      IIwritedb(".");
      IIwritedb(satt[select].a_name);
      IIwritedb(",");
       }
      IIwritedb(satt[select].a_name);
      IIwritedb("=");
      IIwritedb(temp_table1);
      IIwritedb(".");
      IIwritedb(satt[select].a_name);
      IIwritedb(")");
      IIcsrQuery((char *)0);
    } /* IIcsrOpen */

/*&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
  {
    IIsqInit((char *)0);
    IIwritedb("create ");
    IIwritedb(temp_table);
    IIwritedb("(");
    for (i=0;i<n-1;i++){
      IIwritedb(satt[i].a_name);
```

```
        IIwritedb("=");
        if ((strcmp(satt[i].data_type, "image") == 0) ||
                    (strcmp(satt[i].data_type, "sound") == 0) ||
                            (strcmp(satt[i].data_type, "integer") == 0))
          IIwritedb("i4,");
        else
          if (strcmp(satt[i].data_type, "float") == 0)
            IIwritedb("f4,");
          else
            {                                          /* char data_type */
              IIwritedb(satt[i].data_type);
              IIwritedb(",");
            }
      } /* End of for loop i */
    IIwritedb(satt[i].a_name);
    IIwritedb("=");
    if ((strcmp(satt[i].data_type, "image") == 0) ||
              (strcmp(satt[i].data_type, "sound") == 0) ||
                  (strcmp(satt[i].data_type, "integer") == 0))
      IIwritedb("i4");
    else
      if (strcmp(satt[i].data_type, "float") == 0)
       IIwritedb("f4");
      else
       {                                       /* char data_type */
         IIwritedb(satt[i].data_type);
       }
    IIwritedb(")");
    IIsqSync(0,(char *)0);
  }
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/

/* # line 3169 "db.sc" */  /* select */
  {
    IIsqInit((char *)0);
    IIwritedb("retrieve(g=(count(");
    IIwritedb(temp_table);
    IIwritedb(".");
    IIwritedb(satt[0].a_name);
    IIwritedb(")))");
    IIsqRinit((char *)0);
    if (IIerrtest() == 0) {
      if (IInextget() != 0) {
```

```
        IIretdom(1,30,4,&g);
       }
       IIsqFlush((char *)0);
     }
   }


/* # line 3171 "db.sc" */  /* host code */
    if (IIcsrOpen((char *)0,"cursor_output","db2",0,temp_table) != 0) {
      IIwritedb("retrieve(");
      for (select=0;select<n-1;select++) {
      IIwritedb(satt[select].a_name);
      IIwritedb("=");
      IIwritedb(temp_table);
      IIwritedb(".");
      IIwritedb(satt[select].a_name);
      IIwritedb(",");
       }
      IIwritedb(satt[select].a_name);
      IIwritedb("=");
      IIwritedb(temp_table);
      IIwritedb(".");
      IIwritedb(satt[select].a_name);
      IIwritedb(")");
      IIcsrQuery((char *)0);
    }

/*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 */

/* # line 7 "insert.sc" */  /* insert */
  {
    while (look_more == 0) {
      if (IIcsrFetch((char *)0,"cursor_output","db1") != 0) {
       IIsqInit((char *)0);
```

```
IIwritedb("append to ");
IIwritedb(temp_table);
IIwritedb("(");
for (i=0;i<n-1;i++) {
  IIwritedb(satt[i].a_name);
  IIwritedb("=");
  if (strcmp(satt[i].data_type,"c20")==0) {
    IIcsrRet(1,32,0, char_value);
    IIsetdom(1,32,0, char_value);
  }
  if (strcmp(satt[i].data_type,"integer")==0) {
    IIcsrRet(1,30,4,&integer_value);
    IIsetdom(1,30,4,&integer_value);
  }
  if (strcmp(satt[i].data_type,"float")==0) {
    IIcsrRet(1,31,4,&real_value);
    IIsetdom(1,31,4,&real_value);
  }
  if (strcmp(satt[i].data_type,"image")==0) {
    IIcsrRet(1,30,4,&media_value);
    IIsetdom(1,30,4,&media_value);
  }
  if (strcmp(satt[i].data_type,"sound")==0) {
    IIcsrRet(1,30,4,&media1_value);
    IIsetdom(1,30,4,&media1_value);
  }
  IIwritedb(",");
}
IIwritedb(satt[i].a_name);
IIwritedb("=");
if (strcmp(satt[i].data_type,"c20")==0) {
  IIcsrRet(1,32,0, char_value);
  IIsetdom(1,32,0, char_value);
}
if (strcmp(satt[i].data_type,"integer")==0) {
  IIcsrRet(1,30,4,&integer_value);
  IIsetdom(1,30,4,&integer_value);
}
if (strcmp(satt[i].data_type,"float")==0) {
  IIcsrRet(1,31,4,&real_value);
  IIsetdom(1,31,4,&real_value);
}
if (strcmp(satt[i].data_type,"image")==0) {
```

```
        IIcsrRet(1,30,4,&media_value);
        IIsetdom(1,30,4,&media_value);
       }
       if (strcmp(satt[i].data_type,"sound")==0) {
        IIcsrRet(1,30,4,&media1_value);
        IIsetdom(1,30,4,&media1_value);
       }

/*     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 */
       }
       IIwritedb(" )");
       IIsqSync(3,(char *)0);
      } /* IIcsrFetch */
    } /* end while */
  }
  IIcsrClose((char *)0,"cursor_output","db1"); /* close the cursor */
  IIcsrClose((char *)0,"cursor_output","db2"); /* close the cursor */
  return(temp_table);
}
/*********************************************************************
This function takes two temp tables and unions them, puts the result in temp_table
and returns the result to the calling function
*********************************************************************/
union_tables_for_demo(temp_table1, temp_table2, temp_table)
char temp_table1[20];
char temp_table2[20];
char temp_table[20];
{

  int c=0,j=0,k=0,l=0,temp, count;
  int o=0,p=0;
  /*char*/ STR_name char_value[21];
  char file_name[20],a;
  int  integer_value,media_value,found,media1_value;
  float real_value;
   int i=0,select=0;
int g=0;
/*printf("\nNow we are in union_tables_for_nested");*/
/* # line 3169 "db.sc" */  /* select */
```

```
  {
    IIsqInit((char *)0);
    IIwritedb("retrieve(c=(count(");
    IIwritedb(temp_table1);
    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 in temp_table %s",c, temp_table1);*/

/* # line 3171 "db.sc" */  /* host code */
    if (IIcsrOpen((char *)0,"cursor_output","db1",0,temp_table1) != 0) {
      IIwritedb("retrieve(");
      for (select=0;select<n-1;select++) {
      IIwritedb(satt[select].a_name);
      IIwritedb("=");
      IIwritedb(temp_table1);
      IIwritedb(".");
      IIwritedb(satt[select].a_name);
      IIwritedb(",");
       }
      IIwritedb(satt[select].a_name);
      IIwritedb("=");
      IIwritedb(temp_table1);
      IIwritedb(".");
      IIwritedb(satt[select].a_name);
      IIwritedb(")");
      IIcsrQuery((char *)0);
    } /* IIcsrOpen */
/* # line 3169 "db.sc" */  /* select */
  {
    IIsqInit((char *)0);
    IIwritedb("retrieve(o=(count(");
    IIwritedb(temp_table2);
    IIwritedb(".");
```

```
    IIwritedb(satt[0].a_name);
    IIwritedb(")))");
    IIsqRinit((char *)0);
    if (IIerrtest() == 0) {
      if (IInextget() != 0) {
        IIretdom(1,30,4,&o);
      } /* IInextget */
      IIsqFlush((char *)0);
    } /* IIerrtest */
  }
l=0;
/*printf("\nThere are %d records in temp_table %s",o, temp_table1);*/

/* # line 3171 "db.sc" */  /* host code */
    if (IIcsrOpen((char *)0,"cursor_output","db3",0,temp_table2) != 0) {
      IIwritedb("retrieve(");
      for (select=0;select<n-1;select++) {
      IIwritedb(satt[select].a_name);
      IIwritedb("=");
      IIwritedb(temp_table2);
      IIwritedb(".");
      IIwritedb(satt[select].a_name);
      IIwritedb(",");
       }
      IIwritedb(satt[select].a_name);
      IIwritedb("=");
      IIwritedb(temp_table2);
      IIwritedb(".");
      IIwritedb(satt[select].a_name);
      IIwritedb(")");
      IIcsrQuery((char *)0);
    } /* IIcsrOpen */

/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
  {
    IIsqInit((char *)0);
    IIwritedb("create ");
    IIwritedb(temp_table);
    IIwritedb("(");
    for (i=0;i<n-1;i++){
      IIwritedb(satt[i].a_name);
        IIwritedb("=");
        if ((strcmp(satt[i].data_type, "image") == 0) ||
```

```
                (strcmp(satt[i].data_type, "sound") == 0) ||
                        (strcmp(satt[i].data_type, "integer") == 0))
          IIwritedb("i4,");
        else
          if (strcmp(satt[i].data_type, "float") == 0)
            IIwritedb("f4,");
          else
            {                                   /* char data_type */
              IIwritedb(satt[i].data_type);
              IIwritedb(",");
            }
      } /* End of for loop i */
    IIwritedb(satt[i].a_name);
    IIwritedb("=");
    if ((strcmp(satt[i].data_type, "image") == 0) ||
            (strcmp(satt[i].data_type, "sound") == 0) ||
                (strcmp(satt[i].data_type, "integer") == 0))
      IIwritedb("i4");
    else
      if (strcmp(satt[i].data_type, "float") == 0)
       IIwritedb("f4");
      else
       {                                    /* char data_type */
        IIwritedb(satt[i].data_type);
       }
    IIwritedb(")");
    IIsqSync(0,(char *)0);
  }
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/

/* # line 3169 "db.sc" */  /* select */
  {
    IIsqInit((char *)0);
    IIwritedb("retrieve(g=(count(");
    IIwritedb(temp_table);
    IIwritedb(".");
    IIwritedb(satt[0].a_name);
    IIwritedb(")))");
    IIsqRinit((char *)0);
    if (IIerrtest() == 0) {
      if (IInextget() != 0) {
        IIretdom(1,30,4,&g);
      }
```

```
      IIsqFlush((char *)0);
    }
  }


/* # line 3171 "db.sc" */  /* host code */
    if (IIcsrOpen((char *)0,"cursor_output","db2",0,temp_table) != 0) {
      IIwritedb("retrieve(");
      for (select=0;select<n-1;select++) {
      IIwritedb(satt[select].a_name);
      IIwritedb("=");
      IIwritedb(temp_table);
      IIwritedb(".");
      IIwritedb(satt[select].a_name);
      IIwritedb(",");
       }
      IIwritedb(satt[select].a_name);
      IIwritedb("=");
      IIwritedb(temp_table);
      IIwritedb(".");
      IIwritedb(satt[select].a_name);
      IIwritedb(")");
      IIcsrQuery((char *)0);
    }

/*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 */

/* # line 7 "insert.sc" */  /* insert */
  {
    while (look_more == 0) {
      if (IIcsrFetch((char *)0,"cursor_output","db1") != 0) {
      IIsqInit((char *)0);
      IIwritedb("append to ");
      IIwritedb(temp_table);
```

```
IIwritedb("(");
for (i=0;i<n-1;i++) {
  IIwritedb(satt[i].a_name);
  IIwritedb("=");
  if (strcmp(satt[i].data_type,"c20")==0) {
    IIcsrRet(1,32,0, char_value);
    IIsetdom(1,32,0, char_value);
  }
  if (strcmp(satt[i].data_type,"integer")==0) {
    IIcsrRet(1,30,4,&integer_value);
    IIsetdom(1,30,4,&integer_value);
  }
  if (strcmp(satt[i].data_type,"float")==0) {
    IIcsrRet(1,31,4,&real_value);
    IIsetdom(1,31,4,&real_value);
  }
  if (strcmp(satt[i].data_type,"image")==0) {
    IIcsrRet(1,30,4,&media_value);
    IIsetdom(1,30,4,&media_value);
  }
  if (strcmp(satt[i].data_type,"sound")==0) {
    IIcsrRet(1,30,4,&media1_value);
    IIsetdom(1,30,4,&media1_value);
  }
  IIwritedb(",");
}
IIwritedb(satt[i].a_name);
IIwritedb("=");
if (strcmp(satt[i].data_type,"c20")==0) {
  IIcsrRet(1,32,0, char_value);
  IIsetdom(1,32,0, char_value);
}
if (strcmp(satt[i].data_type,"integer")==0) {
  IIcsrRet(1,30,4,&integer_value);
  IIsetdom(1,30,4,&integer_value);
}
if (strcmp(satt[i].data_type,"float")==0) {
  IIcsrRet(1,31,4,&real_value);
  IIsetdom(1,31,4,&real_value);
}
if (strcmp(satt[i].data_type,"image")==0) {
  IIcsrRet(1,30,4,&media_value);
  IIsetdom(1,30,4,&media_value);
```

```
        }
        if (strcmp(satt[i].data_type,"sound")==0) {
          IIcsrRet(1,30,4,&media1_value);
          IIsetdom(1,30,4,&media1_value);
        }
/*      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 */
          }
        IIwritedb(" )");
        IIsqSync(3,(char *)0);
       } /* IIcsrFetch */
     } /* end while */
   }
  IIcsrClose((char *)0,"cursor_output","db2"); /* close the cursor */


/* # line 3171 "db.sc" */  /* host code */
    if (IIcsrOpen((char *)0,"cursor_output","db2",0,temp_table) != 0) {
      IIwritedb("retrieve(");
      for (select=0;select<n-1;select++) {
      IIwritedb(satt[select].a_name);
      IIwritedb("=");
      IIwritedb(temp_table);
      IIwritedb(".");
      IIwritedb(satt[select].a_name);
      IIwritedb(",");
       }
      IIwritedb(satt[select].a_name);
      IIwritedb("=");
      IIwritedb(temp_table);
      IIwritedb(".");
      IIwritedb(satt[select].a_name);
      IIwritedb(")");
      IIcsrQuery((char *)0);
    }

/*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 */

/* # line 7 "insert.sc" */  /* insert */
  {
    while (look_more == 0) {
      if (IIcsrFetch((char *)0,"cursor_output","db3") != 0) {
      IIsqInit((char *)0);
      IIwritedb("append to ");
      IIwritedb(temp_table);
      IIwritedb("(");
      for (i=0;i<n-1;i++) {
        IIwritedb(satt[i].a_name);
        IIwritedb("=");
        if (strcmp(satt[i].data_type,"c20")==0) {
          IIcsrRet(1,32,0, char_value);
          IIsetdom(1,32,0, char_value);
        }
        if (strcmp(satt[i].data_type,"integer")==0) {
          IIcsrRet(1,30,4,&integer_value);
          IIsetdom(1,30,4,&integer_value);
        }
        if (strcmp(satt[i].data_type,"float")==0) {
          IIcsrRet(1,31,4,&real_value);
          IIsetdom(1,31,4,&real_value);
        }
        if (strcmp(satt[i].data_type,"image")==0) {
          IIcsrRet(1,30,4,&media_value);
          IIsetdom(1,30,4,&media_value);
        }
        if (strcmp(satt[i].data_type,"sound")==0) {
          IIcsrRet(1,30,4,&media1_value);
          IIsetdom(1,30,4,&media1_value);
        }
        IIwritedb(",");
      }
      IIwritedb(satt[i].a_name);
      IIwritedb("=");
```

```
        if (strcmp(satt[i].data_type,"c20")==0) {
          IIcsrRet(1,32,0, char_value);
          IIsetdom(1,32,0, char_value);
        }
        if (strcmp(satt[i].data_type,"integer")==0) {
          IIcsrRet(1,30,4,&integer_value);
          IIsetdom(1,30,4,&integer_value);
        }
        if (strcmp(satt[i].data_type,"float")==0) {
          IIcsrRet(1,31,4,&real_value);
          IIsetdom(1,31,4,&real_value);
        }
        if (strcmp(satt[i].data_type,"image")==0) {
          IIcsrRet(1,30,4,&media_value);
          IIsetdom(1,30,4,&media_value);
        }
        if (strcmp(satt[i].data_type,"sound")==0) {
          IIcsrRet(1,30,4,&media1_value);
          IIsetdom(1,30,4,&media1_value);
        }
/*      printf("\n");*/
        IIcsrEFetch((char *)0);   /* fetch the next record to the cursor */
        l++;  /* increment l as the counter */

          if (l==o) {   /* check if no more data to print */
            look_more =1;  /* exit of the loop */
          }
        IIwritedb(" )");
        IIsqSync(3,(char *)0);
       } /* IIcsrFetch */
     } /* end while */
   }

  IIcsrClose((char *)0,"cursor_output","db2"); /* close the cursor */
  IIcsrClose((char *)0,"cursor_output","db1"); /* close the cursor */
  IIcsrClose((char *)0,"cursor_output","db3"); /* close the cursor */
  return(temp_table);
}
/*********************************************************************
This function retrieves the tuples from temp_table1 which do not take place in
temp_table2 and puts the result in temp_table.
*********************************************************************/
minus(temp_table1, temp_table2, temp_table)
```

```
char temp_table1[20];
char temp_table2[20];
char temp_table[20];
{
  int i;
       IIsqInit((char *)0);
       IIwritedb("retrieve into ");
       IIwritedb(temp_table);
       IIwritedb("(");
       for (i=0;i<n-1;i++) {
         IIwritedb(temp_table1);
         IIwritedb(".");
         IIwritedb(satt[i].a_name);
         IIwritedb(",");
       }
       IIwritedb(temp_table1);
       IIwritedb(".");
       IIwritedb(satt[i].a_name);
        IIwritedb(")");

 IIwritedb("where any(");
 IIwritedb(temp_table2);
 IIwritedb(".all by ");
 IIwritedb(temp_table1);
 IIwritedb(".all ");
 IIwritedb(" where(");
 for (i=0;i<n-1;i++) {
   IIwritedb(temp_table1);
   IIwritedb(".");
   IIwritedb(satt[i].a_name);
   IIwritedb("=");
   IIwritedb(temp_table2);
   IIwritedb(".");
   IIwritedb(satt[i].a_name);
   IIwritedb(" and ");
 }
 IIwritedb(temp_table1);
 IIwritedb(".");
 IIwritedb(satt[i].a_name);
 IIwritedb("=");
 IIwritedb(temp_table2);
 IIwritedb(".");
 IIwritedb(satt[i].a_name);
```

```
  IIwritedb(" )");
  IIwritedb(") = 0");
  IIsqSync(0,(char *)0);
  return(temp_table);
}
/*****************************************************************
This function intersects two tables and puts the result in temp_table.
*****************************************************************/
intersect_tables(temp_table1, temp_table2, temp_table)
char temp_table1[20];
char temp_table[20];
{
  int i;
/*      copy_to_file(temp_table1);*/
  IIsqInit((char *)0);
        IIwritedb("retrieve into ");
        IIwritedb(temp_table);
        IIwritedb("(");
        for (i=0;i<n-1;i++) {
          IIwritedb(temp_table1);
          IIwritedb(".");
          IIwritedb(satt[i].a_name);
          IIwritedb(",");
        }
        IIwritedb(temp_table1);
        IIwritedb(".");
        IIwritedb(satt[i].a_name);
         IIwritedb(")");

         IIwritedb(" where(");
        for (i=0;i<n-1;i++) {
        IIwritedb(temp_table1);
        IIwritedb(".");
        IIwritedb(satt[i].a_name);
        IIwritedb("=");
        IIwritedb(temp_table2);
        IIwritedb(".");
        IIwritedb(satt[i].a_name);
        IIwritedb(" and ");
       }
        IIwritedb(temp_table1);
        IIwritedb(".");
        IIwritedb(satt[i].a_name);
```

```
        IIwritedb("=");
        IIwritedb(temp_table2);
        IIwritedb(".");
        IIwritedb(satt[i].a_name);
        IIwritedb(" )");
        IIsqSync(0,(char *)0);
         return(temp_table);
}
/*****************************************************************
This function retrieves the tuples from temp_table1 which are not included in temp2 and
puts the result in temp_table.
*****************************************************************/
temp1_not_in_temp2(temp_table1, temp_table2, temp_table)
char temp_table[20];
char temp_table1[20];
char temp_table2[20];
{
  int i,j;
  j=0;

  printf("\nWe are in table1_NOT_IN_table2 now");

  sqlca.sqlcode = 0;  /* Initialize as error free before access INGRES */
  IIsqInit(&sqlca);
  IIwritedb("retrieve into ");
  IIwritedb(temp_table);
  IIwritedb("(");
  for (i=0;i<n-1;i++) {
    IIwritedb(satt[i].t_name);
    IIwritedb(".");
    IIwritedb(satt[i].a_name);
    IIwritedb(",");
  }
  IIwritedb(satt[i].t_name);
  IIwritedb(".");
  IIwritedb(satt[i].a_name);
  IIwritedb(")");

  IIwritedb("where(any(");
  IIwritedb(temp_table2);
  IIwritedb(".");
  IIwritedb(attribute_for_nested);
/*  IIwritedb(".all by ");*/
```

```
/*  IIwritedb(temp_table1);*/
  IIwritedb(" by ");
  IIwritedb(satt[i].t_name);
  IIwritedb(".all ");
  IIwritedb("where ");
  IIwritedb("(");
  IIwritedb(satt[i].t_name);
/*  IIwritedb(temp_table1);*/
  IIwritedb(".");
  IIwritedb(condition_for_nested);
  IIwritedb("=");
  IIwritedb(temp_table2);
  IIwritedb(".");
  IIwritedb(attribute_for_nested);
  IIwritedb(")");
  IIwritedb(") = 0");
  if (m>1){
    IIwritedb(" and ");
    IIwritedb("(");
    IIwritedb(join_condition);
    IIwritedb(")");
    }
  IIwritedb(")");
  IIsqSync(0,&sqlca);

  if (sqlca.sqlcode != 0){
    printf("\nAn error occurred while accessing the database");
    for (j=j+1; j<m; j++){
      init_buffer(temp_table1,20);
      strcpy(temp_table1, stab[j].t_name);

      sqlca.sqlcode = 0;  /* Initialize as error free before access INGRES */

      IIsqInit(&sqlca);
      IIwritedb("retrieve into ");
      IIwritedb(temp_table);
      IIwritedb("(");
      for (i=0;i<n-1;i++) {
       IIwritedb(satt[i].t_name);
       IIwritedb(".");
       IIwritedb(satt[i].a_name);
       IIwritedb(",");
      }
```

```
        IIwritedb(satt[i].t_name);
        IIwritedb(".");
        IIwritedb(satt[i].a_name);
        IIwritedb(")");

        IIwritedb("where(any(");
        IIwritedb(temp_table2);
        IIwritedb(".");
        IIwritedb(attribute_for_nested);
        IIwritedb(" by ");
/*      IIwritedb(".all by ");*/
        IIwritedb(satt[i].t_name);
/*      IIwritedb(temp_table1);*/
        IIwritedb(".all ");
        IIwritedb("where ");
        IIwritedb("(");
        IIwritedb(satt[i].t_name);
/*      IIwritedb(temp_table1);*/
        IIwritedb(".");
        IIwritedb(condition_for_nested);
        IIwritedb("=");
        IIwritedb(temp_table2);
        IIwritedb(".");
        IIwritedb(attribute_for_nested);
        IIwritedb(")");
        IIwritedb(") = 0");
        if (m>1){
         IIwritedb(" and ");
         IIwritedb("(");
         IIwritedb(join_condition);
         IIwritedb(")");
        }
        IIwritedb(")");
        IIsqSync(0,&sqlca);

      }/* end for j<m */
    }/* end if */

}
/*****************************************************************
This function joins temp1 and temp2 and retrives the tuples from temp1 that takes place
in temp2 and puts the result in temp_table.
*****************************************************************/
```

```
temp1_in_temp2(temp_table1, temp_table2, temp_table)
char temp_table[20];
char temp_table1[20];
char temp_table2[20];
{
  int i,j;
  j=0;

  printf("\nWe are in table1_IN_table2 now");
  sqlca.sqlcode = 0;  /* Initialize as error free before access INGRES */
  IIsqInit(&sqlca);
  IIwritedb("retrieve into ");
  IIwritedb(temp_table);
  IIwritedb("(");
  for (i=0;i<n-1;i++) {
    IIwritedb(satt[i].t_name);
    IIwritedb(".");
    IIwritedb(satt[i].a_name);
    IIwritedb(",");
  }
  IIwritedb(satt[i].t_name);
  IIwritedb(".");
  IIwritedb(satt[i].a_name);
  IIwritedb(")");

  IIwritedb("where(");
  IIwritedb("(");
  IIwritedb(temp_table1);
  IIwritedb(".");
  IIwritedb(condition_for_nested);
  IIwritedb("=");
  IIwritedb(temp_table2);
  IIwritedb(".");
  IIwritedb(attribute_for_nested);
  IIwritedb(")");
  if (m>1){
    IIwritedb(" and ");
    IIwritedb("(");
    IIwritedb(join_condition);
    IIwritedb(")");
  }
  IIwritedb(")");
  IIsqSync(0,&sqlca);
```

```
 if (sqlca.sqlcode != 0){
  for (j=j+1; j<m; j++){
    init_buffer(temp_table1,20);
    strcpy(temp_table1, stab[j].t_name);
    sqlca.sqlcode = 0;  /* Initialize as error free before access INGRES */
    IIsqInit(&sqlca);
    IIwritedb("retrieve into ");
    IIwritedb(temp_table);
    IIwritedb("(");
    for (i=0;i<n-1;i++) {
     IIwritedb(satt[i].t_name);
     IIwritedb(".");
     IIwritedb(satt[i].a_name);
     IIwritedb(",");
     }
    IIwritedb(satt[i].t_name);
    IIwritedb(".");
    IIwritedb(satt[i].a_name);
    IIwritedb(")");

    IIwritedb("where(");
    IIwritedb("(");
    IIwritedb(temp_table1);
    IIwritedb(".");
    IIwritedb(condition_for_nested);
    IIwritedb("=");
    IIwritedb(temp_table2);
    IIwritedb(".");
    IIwritedb(attribute_for_nested);
    IIwritedb(")");
    if (m>1){
     IIwritedb(" and ");
     IIwritedb("(");
     IIwritedb(join_condition);
     IIwritedb(")");
     }
    IIwritedb(")");
    IIsqSync(0,&sqlca);
   }/* end for */
 }/* end if */

}
/*****************************************************************
```

```
This function joins temp1 and temp2 and retrieves the tuples from temp1 that do not take
place in temp2 and puts the result in temp_table.
**********************************************************************/
temp1_not_exists_temp2(temp_table1, temp_table2, temp_table)
char temp_table[20];
char temp_table1[20];
char temp_table2[20];
{
  int i,j;
  j=0;
  printf("\nWe are in table1_not_exists_table2 now");
  sqlca.sqlcode = 0;  /* Initialize as error free before access INGRES */

  IIsqInit(&sqlca);
  IIwritedb("retrieve into ");
  IIwritedb(temp_table);
  IIwritedb("(");
  for (i=0;i<n-1;i++) {
    IIwritedb(satt[i].t_name);
    IIwritedb(".");
    IIwritedb(satt[i].a_name);
    IIwritedb(",");
  }
  IIwritedb(satt[i].t_name);
  IIwritedb(".");
  IIwritedb(satt[i].a_name);
  IIwritedb(")");

  IIwritedb("where(any(");
  IIwritedb(temp_table2);
  IIwritedb(".all by ");
  IIwritedb(satt[i].t_name);
/*  IIwritedb(temp_table1);*/
  IIwritedb(".all ");
  IIwritedb("where ");
  IIwritedb("(");
  IIwritedb(join_for_nested);
  IIwritedb(")");
  IIwritedb(")=0");
  if (m>1){
    IIwritedb(" and ");
    IIwritedb("(");
    IIwritedb(join_condition);
```

```
  IIwritedb(")");
}
IIwritedb(")");

IIsqSync(0,&sqlca);

if (sqlca.sqlcode != 0){
  printf("\nError occurred while accessing the database");
  for (j=j+1; j<m; j++){
    init_buffer(temp_table1,20);
    strcpy(temp_table1, stab[j].t_name);
    sqlca.sqlcode = 0;  /* Initialize as error free before access INGRES */

    IIsqInit(&sqlca);
    IIwritedb(temp_table);
    IIwritedb("(");
    for (i=0;i<n-1;i++) {
     IIwritedb(satt[i].t_name);
     IIwritedb(".");
     IIwritedb(satt[i].a_name);
     IIwritedb(",");
    }
    IIwritedb(satt[i].t_name);
    IIwritedb(".");
    IIwritedb(satt[i].a_name);
    IIwritedb(")");

    IIwritedb("where(any(");
    IIwritedb(temp_table2);
    IIwritedb(".all by ");
    IIwritedb(satt[i].t_name);
    /*IIwritedb(temp_table1);*/
    IIwritedb(".all ");
    IIwritedb("where ");
    IIwritedb("(");
    IIwritedb(join_for_nested);
    IIwritedb(")");
    IIwritedb(")=0");
    if (m>1){
     IIwritedb(" and ");
     IIwritedb("(");
     IIwritedb(join_condition);
     IIwritedb(")");
```

```
      }
      IIwritedb(")");
      IIsqSync(0,&sqlca);

    }/* end j<m */
  }/* end sqlca.sqlcode != 0 */

}
/*****************************************************************
This function retrives the tuples from temp1 that exists in temp2 and puts the
result in temp_table.
*****************************************************************/
temp1_exists_temp2(temp_table1, temp_table2, temp_table)
char temp_table[20];
char temp_table1[20];
char temp_table2[20];
{
  int i,j;
    j=0;

  printf("\nWe are in table1_exists_table2 now");
  sqlca.sqlcode = 0;  /* Initialize as error free before access INGRES */

  IIsqInit(&sqlca);
  IIwritedb("retrieve into ");
  IIwritedb(temp_table);
  IIwritedb("(");
  for (i=0;i<n-1;i++) {
    IIwritedb(satt[i].t_name);
    IIwritedb(".");
    IIwritedb(satt[i].a_name);
    IIwritedb(",");
  }
  IIwritedb(satt[i].t_name);
  IIwritedb(".");
  IIwritedb(satt[i].a_name);
  IIwritedb(")");

  IIwritedb("where(");
  IIwritedb("(");
  IIwritedb(join_for_nested);
  IIwritedb(")");
  if (m>1){
```

```
  IIwritedb(" and ");
  IIwritedb("(");
  IIwritedb(join_condition);
  IIwritedb(")");
 }
IIwritedb(")");
IIsqSync(0,&sqlca);

if (sqlca.sqlcode != 0){
  for (j=j+1; j<m; j++){
    init_buffer(temp_table1,20);
    strcpy(temp_table1, stab[j].t_name);
    sqlca.sqlcode = 0;  /* Initialize as error free before access INGRES */

    IIsqInit(&sqlca);
    IIwritedb("retrieve into ");
    IIwritedb(temp_table);
    IIwritedb("(");
    for (i=0;i<n-1;i++) {
     IIwritedb(satt[i].t_name);
     IIwritedb(".");
     IIwritedb(satt[i].a_name);
     IIwritedb(",");
    }
    IIwritedb(satt[i].t_name);
    IIwritedb(".");
    IIwritedb(satt[i].a_name);
    IIwritedb(")");

    IIwritedb("where(");
    IIwritedb("(");
    IIwritedb(join_for_nested);
    IIwritedb(")");
    if (m>1){
     IIwritedb(" and ");
     IIwritedb("(");
     IIwritedb(join_condition);
     IIwritedb(")");
    }
    IIwritedb(")");
    IIsqSync(0,&sqlca);
  }/* end if j<m */
}/* end for */
```

```
}
/*************************************************
This function calculates the number of tuples retrieved in the result table and prints the
number of tuples.
*************************************************/
void print_count(temp_table, i)
char temp_table[20];
int i;
{
  int t=0;
    {
      IIsqInit((char *)0);
      IIwritedb("retrieve unique(t=(");
      IIwritedb("count");
      IIwritedb("(");
      IIwritedb(temp_table);
      IIwritedb(".");
      IIwritedb(satt[i].a_name);
      IIwritedb(")))");
      IIsqRinit((char *)0);
      if (IIerrtest() == 0) {
       if (IInextget() != 0) {
         IIretdom(1,30,4,&t);
       } /* IInextget */
       IIsqFlush((char *)0);
      } /* IIerrtest */
    }
  printf("COUNT(%s) = %d  ",satt[i].a_name, t);
}
/*************************************************
This function calculates the sum of a cloumn retrieved in the result table and prints the
sum.
*************************************************/
void print_sum(temp_table, i)
char temp_table[20];
int i;
{
  int t=0;
    {
      IIsqInit((char *)0);
      IIwritedb("retrieve unique(t=(");
      IIwritedb("sum");
      IIwritedb("(");
```

```
      IIwritedb(temp_table);
      IIwritedb(".");
      IIwritedb(satt[i].a_name);
      IIwritedb(")))");
      IIsqRinit((char *)0);
      if (IIerrtest() == 0) {
       if (IInextget() != 0) {
         IIretdom(1,30,4,&t);
       } /* IInextget */
       IIsqFlush((char *)0);
      } /* IIerrtest */
    }
  printf("SUM(%s) = %d  ",satt[i].a_name, t);
}
/*************************************************
This function calculates the average of an attribute of a tuple retrieved in the result table
and prints the average.
*************************************************/
void print_avg(temp_table, i)
char temp_table[20];
int i;
{
  int t=0;
    {
      IIsqInit((char *)0);
      IIwritedb("retrieve unique(t=(");
      IIwritedb("avg");
      IIwritedb("(");
      IIwritedb(temp_table);
      IIwritedb(".");
      IIwritedb(satt[i].a_name);
      IIwritedb(")))");
      IIsqRinit((char *)0);
      if (IIerrtest() == 0) {
       if (IInextget() != 0) {
         IIretdom(1,30,4,&t);
       } /* IInextget */
       IIsqFlush((char *)0);
      } /* IIerrtest */
    }
  printf("AVG(%s) = %d",satt[i].a_name, t);
}
/*************************************************
```

```
This function finds the max of a coloumn of a tuple in the temp_table and prints the max.
**************************************************/
void print_max(temp_table, i)
char temp_table[20];
int i;
{
  int t=0;
    {
      IIsqInit((char *)0);
      IIwritedb("retrieve unique(t=(");
      IIwritedb("max");
      IIwritedb("(");
      IIwritedb(temp_table);
      IIwritedb(".");
      IIwritedb(satt[i].a_name);
      IIwritedb(")))");
      IIsqRinit((char *)0);
      if (IIerrtest() == 0) {
       if (IInextget() != 0) {
         IIretdom(1,30,4,&t);
       } /* IInextget */
       IIsqFlush((char *)0);
      } /* IIerrtest */
    }
  printf("MAX(%s) = %d  ",satt[i].a_name, t);
}
/**************************************************
This function calculates the min of an attribute of a tuple retrieved in the temp_table and
prints the min.
**************************************************/
void print_min(temp_table, i)
char temp_table[20];
int i;
{
  int t=0;
    {
      IIsqInit((char *)0);
      IIwritedb("retrieve unique(t=(");
      IIwritedb("min");
      IIwritedb("(");
      IIwritedb(temp_table);
      IIwritedb(".");
      IIwritedb(satt[i].a_name);
```

```
      IIwritedb(")))");
      IIsqRinit((char *)0);
      if (IIerrtest() == 0) {
       if (IInextget() != 0) {
         IIretdom(1,30,4,&t);
       } /* IInextget */
       IIsqFlush((char *)0);
      } /* IIerrtest */
    }
  printf("MIN(%s) = %d  ",satt[i].a_name, t);
}
/*************************************************************************
This function checks the aggregate type in the struct satt and calls the appropriate
function.
*************************************************************************/
print_aggregates(temp_table)
char temp_table[20];
{
  int v;
  for(v=0; v<n; v++){
    if (satt[v].aggregate_type==1)
      print_count(temp_table, v);
    if (satt[v].aggregate_type==2)
      print_sum(temp_table, v);
    if (satt[v].aggregate_type==3)
      print_avg(temp_table, v);
    if (satt[v].aggregate_type==4)
      print_max(temp_table, v);
    if (satt[v].aggregate_type==5)
      print_min(temp_table, v);
  }
}
/*************************************************************************
This function prints the tuples retrived in the result table.
*************************************************************************/
print_result_table(temp_table,flag,c)
char temp_table[20];
int flag;
int c;
{
  int v;
  int j=0,k=0,l=0,temp,select=0;
  char char_value[21],a, Ans;
```

```
  char file_name[20];
  int  integer_value,media_value,found,media1_value;
  float real_value;
  int record_id;
  int i=0;
  c=0;
/* # line 3169 "db.sc" */  /* select */
  {
    IIsqInit((char *)0);
    IIwritedb("retrieve unique(c=(count(");
    IIwritedb(temp_table);
    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;
if (flag==FALSE){
  printf("\nThere are %d records that match the query",c);
  /*--------*/
  if (c==0) {
    printf("\nPress ENTER to continue...");
    a=getchar();
    return;
  }
}
/* # line 3171 "db.sc" */  /* host code */
    if (IIcsrOpen((char *)0,"cursor_output","db1",0,temp_table) != 0) {
      IIwritedb("retrieve (");
      for (select=0;select<n-1;select++) {
      IIwritedb(satt[select].a_name);
      IIwritedb("=");
      IIwritedb(temp_table);
      IIwritedb(".");
      IIwritedb(satt[select].a_name);
      IIwritedb(",");
       }
```

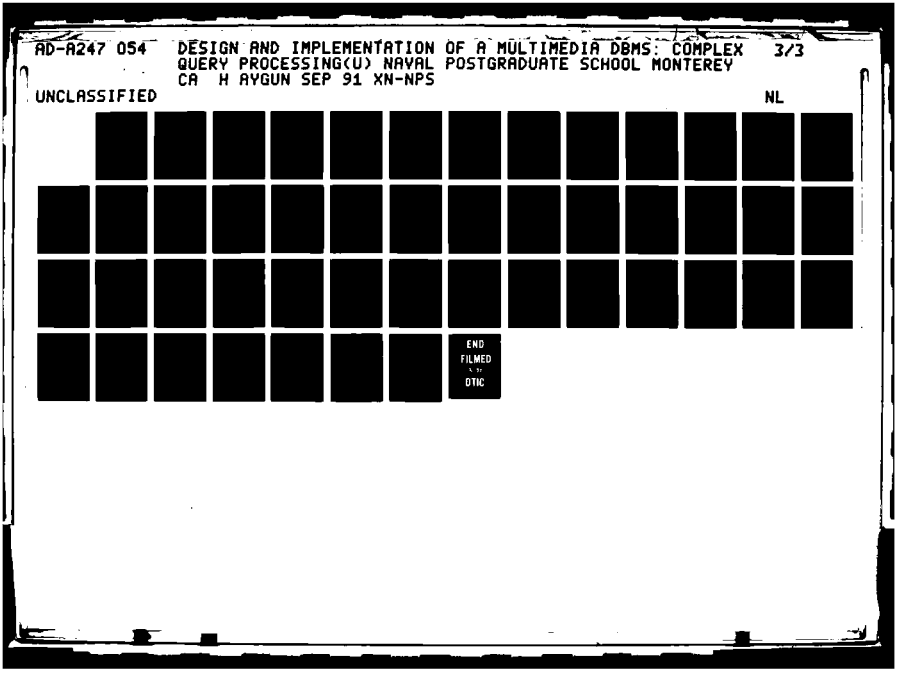
```
      IIwritedb(satt[select].a_name);
      IIwritedb("=");
      IIwritedb(temp_table);
      IIwritedb(".");
      IIwritedb(satt[select].a_name);
      IIwritedb(")");
      IIcsrQuery((char *)0);
    } /* IIcsrOpen */
printf("\n");
look_more=0;
l=0;
if (c==0) {
      look_more=1;
      }
/* Fetch 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);
          if (satt[i].aggregate_type==0)
            printf("%s : %s",satt[i].a_name,char_value);

          }
        if (strcmp(satt[i].data_type,"integer")==0) {
          IIcsrRet(1,30,4,&integer_value);
          if (satt[i].aggregate_type==0)
            printf("%s : %d  ",satt[i].a_name,integer_value);
          }
        if (strcmp(satt[i].data_type,"float")==0) {
          IIcsrRet(1,31,4,&real_value);
          if (satt[i].aggregate_type==0)
            printf("%s : %8.2f  ",satt[i].a_name,real_value);
          }
        if (strcmp(satt[i].data_type,"image")==0) {
          IIcsrRet(1,30,4,&media_value);
          if (satt[i].aggregate_type==0)
            printf("%s id is %d  ",satt[i].a_name,media_value);
          }
        if (strcmp(satt[i].data_type,"sound")==0) {
```

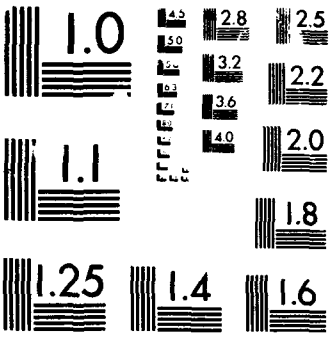
```
            IIcsrRet(1,30,4,&media1_value);
            if (satt[i].aggregate_type==0)
              printf("%s id is %d",satt[i].a_name,media1_value);
            }
          } /* end for select < n*/
            IIcsrEFetch((char *)0);   /* fetch the next record to the cursor */
            l++;  /* increment l as the counter */
            if (l==c) {   /* check if no more data to print */
              look_more =1;  /* exit of the loop */
            }
            print_aggregates(temp_table);
            printf("\n");
            } /* IIcsrFetch */
  } /* end while */
  IIcsrClose((char *)0,"cursor_output","db1"); /* close the cursor */
  if (flag==FALSE){
    printf("Press ENTER to continue ..");
    a= getchar();
  }
/* this for the check for the media selection */
  if (c==0) {
      i=9999;
      }

/* if there are some aggregate functions print their results */
/*  print_aggregates(temp_table);
  printf("\nPress ENTER to continue ..");
  a= getchar();*/
  return(c);
}
/*************************************************************************
This function gets the image id of a tuple in the result table.
*************************************************************************/
get_image_id(r,image_id)
int r;
int image_id;
{
  int sound_id;
  int entry;
  int desired_tuple;
  char c_temp[60];
  int count=0;
  int j=0,k=0,l=0,temp;
```

```
  char char_value[21],a;
  char file_name[20];
  int img_value, snd_value;
  int  integer_value,media_value,found,media1_value;
  float real_value;
   int i=0,select=0;
  int g=0;
  int d;
  i_value[i_index]=0;
  desired_tuple=r;
/* # line 3169 "db.sc" */  /* select */
  {
   IIsqInit((char *)0);
   IIwritedb("retrieve(g=(count(");
   IIwritedb(temp_table);
   IIwritedb(".");
   IIwritedb(satt[0].a_name);
   IIwritedb(")))");
   IIsqRinit((char *)0);
   if (IIerrtest() == 0) {
    if (IInextget() != 0) {
     IIretdom(1,30,4,&g);
    } /* IInextget */
    IIsqFlush((char *)0);
   } /* IIerrtest */
  }
l=0;
if (g==0) {
      printf("\nPress ENTER to continue...");
      a=getchar();
      return;
      }
/* # line 3171 "db.sc" */  /* host code */
   if (IIcsrOpen((char *)0,"cursor_output","db2",0,temp_table) != 0) {
    IIwritedb("retrieve(");
    for (select=0;select<n-1;select++) {
    IIwritedb(satt[select].a_name);
    IIwritedb("=");
    IIwritedb(temp_table);
    IIwritedb(".");
    IIwritedb(satt[select].a_name);
    IIwritedb(",");
     }
```

```
    IIwritedb(satt[select].a_name);
    IIwritedb("=");
    IIwritedb(temp_table);
    IIwritedb(".");
    IIwritedb(satt[select].a_name);
    IIwritedb(")");
    IIcsrQuery((char *)0);
  } /* IIcsrOpen */
printf("\n");
look_more=0;
l=0;
if (g==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","db2") != 0) {
    if (desired_tuple == l){
     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);
         if (desired_tuple == l){
           printf("%s :  %s",satt[i].a_name,char_value);
         }
       }
       if (strcmp(satt[i].data_type,"integer")==0) {
         IIcsrRet(1,30,4,&integer_value);
         if (desired_tuple == l){
           printf("%s :  %d  ",satt[i].a_name,integer_value);
         }
       }
       if (strcmp(satt[i].data_type,"float")==0) {
         IIcsrRet(1,31,4,&real_value);
         if (desired_tuple == l){
           printf("%s :  %8.2f  ",satt[i].a_name,real_value);
         }
       }
       if (strcmp(satt[i].data_type,"image")==0) {
```

```
          IIcsrRet(1,30,4,&media_value);
          if (desired_tuple == 1){
            image_id=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);
          if (desired_tuple == 1){
/*          sound_id=media1_value;*/
            printf("%s %d",satt[i].a_name,media1_value);
          }
        }
      }/*end for select n*/
     IIcsrEFetch((char *)0);   /* fetch the next record to the cursor */
     l++;  /* increment l as the counter */
     if (l==g) {   /* check if no more data to print */
      look_more =1;  /* exit of the loop */
     }
    } /* IIcsrFetch */
  } /* end while */

  IIcsrClose((char *)0,"cursor_output","db2"); /* close the cursor */

  printf("\nPress ENTER to continue ..");
  a= getchar();
  return(image_id);
}
/*****************************************************************************
This function gets the sound id of a tuple in the result table.
*****************************************************************************/
get_sound_id(r,sound_id)
int r;
int sound_id;
{
  int image_id;
  int entry;
  int desired_tuple;
  char c_temp[60];
  int count=0;
  int j=0,k=0,l=0,temp;
  char char_value[21],a;
  char file_name[20];
```


AD-A247 054
DESIGN AND IMPLEMENTATION OF A MULTIMEDIA DBMS: COMPLEX QUERY PROCESSING(U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA H AYGUN SEP 91 XN-NPS
3/3
UNCLASSIFIED
NL
END
FILMED
DTIC



```
  int img_value, snd_value;
  int  integer_value,media_value,found,media1_value;
  float real_value;
   int i=0,select=0;
  int g=0;
  int d;
  i_value[i_index]=0;
  desired_tuple=r;
/* # line 3169 "db.sc" */  /* select */
  {
    IIsqInit((char *)0);
    IIwritedb("retrieve(g=(count(");
    IIwritedb(temp_table);
    IIwritedb(".");
    IIwritedb(satt[0].a_name);
    IIwritedb(")))");
    IIsqRinit((char *)0);
    if (IIerrtest() == 0) {
      if (IInextget() != 0) {
        IIretdom(1,30,4,&g);
      } /* IInextget */
      IIsqFlush((char *)0);
    } /* IIerrtest */
  }
l=0;
if (g==0) {
       printf("\nPress ENTER to continue...");
       a=getchar();
       return;
       }
/* # line 3171 "db.sc" */  /* host code */
    if (IIcsrOpen((char *)0,"cursor_output","db2",0,temp_table) != 0) {
      IIwritedb("retrieve(");
      for (select=0;select<n-1;select++) {
      IIwritedb(satt[select].a_name);
      IIwritedb("=");
      IIwritedb(temp_table);
      IIwritedb(".");
      IIwritedb(satt[select].a_name);
      IIwritedb(",");
       }
      IIwritedb(satt[select].a_name);
      IIwritedb("=");
```

```
    IIwritedb(temp_table);
    IIwritedb(".");
    IIwritedb(satt[select].a_name);
    IIwritedb(")");
    IIcsrQuery((char *)0);
  } /* IIcsrOpen */
printf("\n");
look_more=0;
l=0;
if (g==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","db2") != 0) {
    if (desired_tuple == l){
     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);
         if (desired_tuple == l){
           printf("%s :  %s",satt[i].a_name,char_value);
         }
       }
       if (strcmp(satt[i].data_type,"integer")==0) {
         IIcsrRet(1,30,4,&integer_value);
         if (desired_tuple == l){
           printf("%s :  %d  ",satt[i].a_name,integer_value);
         }
       }
       if (strcmp(satt[i].data_type,"float")==0) {
         IIcsrRet(1,31,4,&real_value);
         if (desired_tuple == l){
           printf("%s :  %8.2f  ",satt[i].a_name,real_value);
         }
       }
       if (strcmp(satt[i].data_type,"image")==0) {
         IIcsrRet(1,30,4,&media_value);
         if (desired_tuple == l){
```

```
/*          image_id=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);
           if (desired_tuple == 1){
            sound_id=media1_value;
            printf("%s %d",satt[i].a_name,media1_value);
           }
          }
        }/*end for select n*/
      IIcsrEFetch((char *)0);   /* fetch the next record to the cursor */
      l++;  /* increment l as the counter */
      if (l==g) {    /* check if no more data to print */
       look_more =1;  /* exit of the loop */
      }
     } /* IIcsrFetch */
   } /* end while */

   IIcsrClose((char *)0,"cursor_output","db2"); /* close the cursor */
   return(sound_id);
}

/*************************************************************************
This function calls the function print_result_table and then queries the user if he wants
to display any media data.
*************************************************************************/
ql_printdata(temp_table)
char temp_table[20];

{
  int image_id=0;
  int sound_id=0;
  int c=0,j=0,k=0,l=0,temp,select=0;
  char char_value[21],a, Ans;
  char file_name[20];
  int  integer_value,media_value,found,media1_value;
  float real_value;
  int record_id, flag=FALSE;
  int i=0;
  c=print_result_table(temp_table, flag, c);
  flag=TRUE;
```

```
for (k=0;k<n;k++) {
  if ((strcmp(satt[k].data_type,"image")==0)||(strcmp(satt[k].data_type,"sound")==0)) {
    if (strcmp(satt[k].data_type,"image")==0)
      printf("\nDo you want to display any media data ? (y/n)");
    if (strcmp(satt[k].data_type,"sound")==0)
      printf("\nDo you want to display any media data ? (y/n)");
    Ans=yes_no_answer();
    if ((Ans==121)||(Ans==89)){
  for (k=0;k<n;k++) {
    if (strcmp(satt[k].data_type,"image")==0) {
  Ans =121;
  while ((Ans == 121) || (Ans == 89)){
    if (c>1){
      printf("\n\nWhich tuple's image do you want to see? (enter record id) :");
      scanf("%d", &record_id);
      getchar();
      printf("record_id --> %d", record_id);
    }
    if (c==1)
      record_id=1;
    if (c==0)
      goto final;
    j = record_id - 1;
    image_id=get_image_id(j,image_id);
    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();
       table_cursor = table_entry;
       strcpy(media_name,satt[i].a_name);
       get_media_name();
       display_photo(i,j,temp_table,image_id);
      }
    }
    printf("\nDo you want to see more image data ? (Y/N) :");
    Ans=yes_no_answer();
    if ((Ans==121)||(Ans==89))
      print_result_table(temp_table,flag);
    if ((Ans==110)||(Ans==78))
      goto next;
  }
}
  }
```

```
next:
 for (k=0;k<n;k++) {
   if (strcmp(satt[k].data_type,"sound")==0) {

 Ans =121;
 while ((Ans == 121) || (Ans == 89)){
   print_result_table(temp_table, flag);
   if (c>1){
     printf("\nWhich tuple's sound do you want to hear? (enter record id) :");
     scanf("%d", &record_id);
   }
   if (c==1){
     record_id=1;
   }
   if (c==0)
     goto final;
   j = record_id - 1;
   sound_id=get_sound_id(j,sound_id);
   for (i=0;i<n;i++) {
     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,temp_table, sound_id);
     }
   }
   printf("\nDo you want to hear more sound data ? (Y/N) :");
   Ans=yes_no_answer();
 }
}
  }
}/* end if ans=121 (the one at the top) */
else
  k=900;
}/*end if strcmp(datatype=image or sound) */
}/* end for k<n (top one ) */
  /*printf("\n");*/
/* Drop table result after finished print  */
/*drop_table(temp_table);*/
 final:
```

```
drop_temp_media_tables();
}
/*********************************************
This function drops a table in INGRES.
*********************************************/
drop_table(table_name)
char table_name[20];
{
  {
    IIsqInit((char *)0);
    IIwritedb("destroy ");
    IIwritedb(table_name);
    IIsqSync(0,(char *)0);
  }
}
/***********************
This function initializes an array upto size 100.
***********************/
init_buffer(buffer,j)
char buffer[100];
int j;
{
  int i;
  for (i=0;i<j;i++) {
    buffer[i] = '\0';
  }
}
/**********************
This function drops the temporary media tables used to hold the intermediate results of
a query.
**********************/
drop_temp_media_tables()
{
  int k;
  char l[5];
  char tempstring[100];
  for (k=0; k<10; k++){
    strcpy(tempstring, "p");
    inttostr(k,l);
    strcat(tempstring,l);
    IIsqInit((char *)0);
    IIwritedb("destroy ");
    IIwritedb(tempstring);
```

```
    IIsqSync(0,(char *)0);
    init_buffer(tempstring,100);
    init_buffer(1,5);
  }
}

/*****************************************
This function asks the user to enter a join condition.
*****************************************/
void help_join()
{
  int i=0;
  if (m > 1) {
    strcpy(join_condition,"?");
    while (strcmp(join_condition,"?")==0) {
      printf("\nPlease enter your join condition\n(<?> for help!) :");
      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 */
      } /* end if need help for join */
    }/*end while*/
  } /* end if more than 1 table select */
}
/*************************************
This user asks the user to enter three temp table names for intersection.
*************************************/
char get_temp_table_names_for_intersection(temp_table1,temp_table2,temp_table)
char temp_table1[20];
char temp_table2[20];
char temp_table[20];
{
  printf("\nEnter first temp table name :");
  gets(buff);
  strcpy(temp_table1, buff);
  init_buffer(buff,100);
  printf("\nEnter second temp table name :");
  gets(buff);
  strcpy(temp_table2, buff);
  init_buffer(buff,100);
  printf("\nEnter another temporary table name to hold the result :");
```

```
  gets(buff);
  strcpy(temp_table, buff);
  init_buffer(buff,100);

  return(temp_table1, temp_table2, temp_table);

}
/*************************************
This function shows the intersect/union/minus menu.
*************************************/
char intersect_union_menu(answer)
char answer;
{
  answer = '?';
/*   while (!( '0'<= answer && answer <= '3'))
   (*/
      clr_scr();
       printf("\nIf you want to intersect / union / minus any two temporary tables:\n");
      printf("\t==========================\n");
      printf("\n\t1. INTERSECT two tables");
      printf("\n\t2. UNION two tables");
      printf("\n\t3. MINUS");
      printf("\n\t0. Quit");
      printf("\n\t==========================\n");
      printf("\n\tSelect your choice :: ");
      answer = getchar();
      while ((c = getchar()) != '\n')
      ; /* Not return do nothing */
/*   } */
   return (answer);

}
/*************************************************************
This function asks the user if he wants to union/intersect/minus any  two tables and p[uts
the result in temp_table.
*************************************************************/
query_for_intersect_union(choice,temp_table1,temp_table2,temp_table)
char choice;
char temp_table1[20];
char temp_table2[20];
char temp_table[20];
{
   choice = '?';
```

```
  clr_scr();
  while (choice != '0')
  {
    choice = intersect_union_menu(choice);/* print the choice for user select on screen
*/
    switch(choice)        /* User select case */
    {
      case '1' :          /* create table */
            clr_scr();
              printf("\nYour Selection is INTERSECT");
              printf("\nHit Return to continue! (Any other key to QUIT!)");
              if (getchar() != '\n')
                {
                  getchar();  /* To let next getchar() work well */
                  break;
                }
            get_temp_table_names_for_intersection(temp_table1,   temp_table2,
temp_table);
            printf("\n*** The result of the INTERSECTION will be kept in
temp_table*** %s ***\n", temp_table);
            intersect_tables(temp_table1, temp_table2, temp_table);
            ql_printdata(temp_table);
            break;
      case '2' :
            clr_scr();
              printf("\nYour Selection is UNION");
              printf("\nHit Return to continue! (Any other key to QUIT!)");
              if (getchar() != '\n')
                {
                  getchar();  /* To let next getchar() work well */
                  break;
                }
            get_temp_table_names_for_intersection(temp_table1,   temp_table2,
temp_table);
            printf("\n*** The result of the UNION will be kept in temp_table*** %s
***\n", temp_table);
            union_tables_for_demo(temp_table1, temp_table2, temp_table);
            ql_printdata(temp_table);
            break;
      case '3' :          /* create table */
            clr_scr();
              printf("\nYour Selection is MINUS");
              printf("\nHit Return to continue! (Any other key to QUIT!)");
```

```
                if (getchar() != '\n')
                  {
                    getchar();  /* To let next getchar() work well */
                    break;
                  }
              get_temp_table_names_for_intersection(temp_table1,   temp_table2,
temp_table);
              printf("\n*** The result of the MINUS will be kept in temp_table*** %s
***\n", temp_table);
              minus(temp_table1, temp_table2, temp_table);
              ql_printdata(temp_table);
              break;

        case '0' :
              clr_scr();
                printf("\nHit Return to continue! (Any other key to QUIT!)");
                if (getchar() != '\n')
                  {
                    getchar();  /* To let next getchar() work well */
                    break;
                  }
              break;
      }  /* End of switch */
    }  /* End of while choice != '0' */
   return(choice);
   return(temp_table1);
   return(temp_table2);
   return(temp_table);
}
/*************************************
This function displays the Retrieval operations menu
*************************************/
char show_utility_menu(answer)
char answer;
{
  answer = '?';
/*   while (!( '0'<= answer && answer <= '4'))
    {*/
      clr_scr();
      printf("\n\tRetrieval Operations Menu\n");
      printf("\t=========================\n");
      printf("\n\t0. Simple Condition");
      printf("\n\t1. table1 where EXISTS table2");
```

```
      printf("\n\t2. table1 where NOT EXISTS table2");
      printf("\n\t3. table1 IN table2");
      printf("\n\t4. table1 NOT IN table2");
      printf("\n\t=================================\n");
      printf("\n\tSelect your choice :: ");
      answer = getchar();
      while ((c = getchar()) != '\n')
      ; /* Not return do nothing */
/*    } */
   return (answer);

}
/*************************************************************
Ths function calls the function show_utility_menu and calls other functions to process the
user's choice.
*************************************************************/
utility_menu(choice,temp_table1,temp_table2,temp_table)
char choice;
char temp_table1[20];
char temp_table2[20];
char temp_table[20];
{
   choice = '?';
   clr_scr();

/*   while (choice != '0')
   {*/
      choice = show_utility_menu(choice);/* print the choice for user select on screen */
      switch(choice)        /* User select case */
      {
        case '1' :          /* create table */
              clr_scr();
                printf("\nYour Selection is table1 where EXISTS table2");
                printf("\nHit Return to continue! (Any other key to QUIT!)");
                if (getchar() != '\n')
                  {
                    getchar();  /* To let next getchar() work well */
                    break;
                  }
              printf("\nEnter the temp table name related to EXISTS :");
              gets(buff);
              strcpy(temp_table2, buff);
              init_buffer(buff,100);
```

```
        printf("\nPlease enter your join condition\nbetween ");
        if (m==1)
          printf("%s and ", temp_table1);
        if (m>1)
          printf("the appropriate table and ");
        printf("** %s ** :", temp_table2);
        gets(buff);
        strcpy(join_for_nested, buff);
        init_buffer(buff,100);
        break;
case '2' :
        clr_scr();
          printf("\nYour Selection is table1 where NOT EXISTS table2");
          printf("\nHit Return to continue! (Any other key to QUIT!)");
          if (getchar() != '\n')
            {
              getchar();  /* To let next getchar() work well */
              break;
            }
        printf("\nEnter the temp table name related to NOT EXISTS :");
        gets(buff);
        strcpy(temp_table2, buff);
        init_buffer(buff,100);
        printf("\nPlease enter your join condition\nbetween ");
        if (m==1)
        printf("%s and ", temp_table1);
        if (m>1)
          printf("the appropriate table and ");
        printf("** %s ** :", temp_table2);
        gets(buff);
        strcpy(join_for_nested, buff);
        init_buffer(buff,100);
        break;
case '3' :
        clr_scr();
          printf("\nYour Selection is table1 IN table2");
          printf("\nHit Return to continue! (Any other key to QUIT!)");
          if (getchar() != '\n')
            {
              getchar();  /* To let next getchar() work well */
              break;
            }
        printf("\nEnter the temp table name related to IN :");
```

```
        gets(buff);
        strcpy(temp_table2, buff);
        init_buffer(buff,100);

        printf("\n\nEnter attribute for ");
        if (m==1)
        printf("table %s", temp_table1);
        if (m>1)
          printf("the appropriate table");
        printf(" for condition of IN :");
        gets(buff);
        strcpy(condition_for_nested, buff);
        init_buffer(buff,100);

        printf("\n\nTable ** %s **", temp_table2);
        printf("\nSELECT ATTRIBUTE (only one attribute!) :");
        gets(buff);
        strcpy(attribute_for_nested, buff);
        init_buffer(buff,100);
        break;
case '4' :
        clr_scr();
          printf("\nYour Selection is table1 NOT IN table2");
          printf("\nHit Return to continue! (Any other key to QUIT!)");
          if (getchar() != '\n')
            {
              getchar();  /* To let next getchar() work well */
              break;
            }
        printf("\nEnter the temp table name related to NOT IN :");
        gets(buff);
        strcpy(temp_table2, buff);
        init_buffer(buff,100);

        printf("\n\nEnter attribute for ");
        if (m==1)
        printf("table %s", temp_table1);
        if (m>1)
          printf("the appropriate table");
        printf(" for condition of NOT IN :");
        gets(buff);
        strcpy(condition_for_nested, buff);
        init_buffer(buff,100);
```

```
            printf("\n\nTable ** %s **", temp_table2);
            printf("\nSELECT ATTRIBUTE (only one attribute!) :");
            gets(buff);
            strcpy(attribute_for_nested, buff);
            init_buffer(buff,100);
            break;
        case '0' :
            clr_scr();
              printf("\nYour Selection is NORMAL RETRIEVAL");
              printf("\nHit Return to continue! (Any other key to QUIT!)");
              if (getchar() != '\n')
                {
                  getchar();  /* To let next getchar() work well */
                  break;
                }
            break;
      } /* End of switch */
  /* }*/  /* End of while choice != '0' */

   return(choice);
   return(temp_table1);
   return(temp_table2);
   return(temp_table);
}
/************************************
This function checks if any attributes with aggregate functions exist in the attributes
entered by the user.
************************************/
char check_aggregate(buffer, tmp, aggregate_found)
char buffer[13];
char tmp[3];
{
  int i = 0;
  int jj = 0;
  for (jj=0;jj<3;jj++){
    if (buffer[i]==40){
/*      tmp[jj]='\0';*/
      jj=1000;
    }
    else{
      tmp[jj]=buffer[i];
    }
```

```
    i++;
  } /* end for jj < 3 */
  tmp[3]='\0';
                                                          i                    f
((strcmp(tmp,"cnt")==0)||(strcmp(tmp,"sum")==0)||(strcmp(tmp,"avg")==0)||(strcmp(tmp
,"min")==0)||(strcmp(tmp,"max")==0)){
    aggregate_found=TRUE;
  }
  return(aggregate_found);
}
/*****************************************
When there is an aggregate function among the attributes entered by the user, this
function separates the attribute from the aggregate part.
*****************************************/
char get_attribute(buffer, attribute)
char buffer[13];
char attribute[13];
{
  int i = 4;
  int j;
  for (j=0;j<13;j++){
    if (buffer[i]==41){
      attribute[j]= '\0';
      j=100;
    }
    else{
      attribute[j]=buffer[i];
    }
    i=i+1;
  } /* end for j < 13 */
  return(attribute);
}
/*****************************************
When mod is modify mode (MOD_MODE) this function is the main function calling
other other functions to delete the tuples from the related media tables.
*****************************************/
void delete_for_modify(r)
int r;
{
    int j=0,k=0,l=0,temp;
    char char_value[21],a;
    char file_name[20];
    int  integer_value,media_value,found,media1_value;
```

```
int im_value, so_value;
int desired_tuple;
float real_value;
int i=0,select=0;
int c=0;
desired_tuple=r;
printf("\nTuple # %d is being deleted now ...", desired_tuple+1);
sleep(2);
/* # line 3169 "db.sc" */       /* select */
{
      IIsqInit((char *)0);
     IIwritedb("retrieve unique(c=(count(");
      IIwritedb(temp_table);
      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;
 if (c==0) {
         printf("\nPress ENTER to continue...");
               a=getchar();
               return;
       } /*********/
/* # line 3171 "db.sc" */     /* host code */
   if (IIcsrOpen((char *)0,"cursor_output","db1",0,temp_table) != 0) {
    IIwritedb("retrieve (");
    for (select=0;select<n-1;select++) {
      IIwritedb(satt[select].a_name);
      IIwritedb("=");
      IIwritedb(temp_table);
      IIwritedb(".");
      IIwritedb(satt[select].a_name);
      IIwritedb(",");
    }
    IIwritedb(satt[select].a_name);
    IIwritedb("=");
```

```
      IIwritedb(temp_table);
      IIwritedb(".");
      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 temp_tablerelation which is the intermediate table
       hold the temp_tablefrom 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) {
       if (desired_tuple == l){
         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);
            if (desired_tuple == l)
            printf("%s :  %s",satt[i].a_name,char_value);
          }
          if (strcmp(satt[i].data_type,"integer")==0) {
            IIcsrRet(1,30,4,&integer_value);
            if (desired_tuple == l)
            printf("%s :  %d  ",satt[i].a_name,integer_value);
          }
          if (strcmp(satt[i].data_type,"float")==0) {
            IIcsrRet(1,31,4,&real_value);
            if (desired_tuple == l)
            printf("%s :  %8.2f  ",satt[i].a_name,real_value);
          }
          if (strcmp(satt[i].data_type,"image")==0) {
            IIcsrRet(1,30,4,&media_value);
            if (desired_tuple == l){
            im_value=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);
            if (desired_tuple == 1){
            so_value=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 1 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 ENTER to continue ..");
  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) {
     if (image_flag==TRUE){
       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();
       printf("\nThe media data from the media table *** %s *** is being deleted
now...", media_name);
       sleep(4);
       mod_get_rid_image(i, im_value);
     }
    }
    if (strcmp(satt[i].data_type,"sound")==0) {
     if (sound_flag==TRUE){
       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();
```

```
        printf("\nThe media data from the media table *** %s *** is being deleted
now...", media_name);
        sleep(4);
        mod_get_rid_sound(i, so_value);
       }
      }
    } /* end for select < n*/
      printf("\n");
 }
/***********************************************
When mode is MODIFY, this function gets sound file attributes from the related media
table.
***********************************************/
get_snd_file_atts(media_name, i, value)
STR_name media_name;
int i;
int value;
{
  int entry;
  char sound_value[20];


 int att_cursor;
  int desired_tupleno;
  char query_phrase[DESCRLEN+1],
       in_phrase[DESCRLEN+1];
  int j=0, k, c, pid, query_err, query_len, in_len, f_flag,look_more=0;

  char ISfn5[FILENAMELEN+1];
  char ISdescr1[DESCRLEN+1];
  int  ISerror;
  STR_path file_name;
  STR_descrp nothing;
  char temp_file[100];  /* Declare more to avoid bus error */
  int show_pid, wait_pid;
  union wait status;
  int sid = 0;
  int pp=0;
  int qq=0;
  int res=0;
  int sz=0;
  int s_rate=0;
  int enc;
```

```
int dur=0;


inttostr(value, sound_value);
{
 IIsqInit ((char *)0);
 IIwritedb("retrieve unique(pp=(count(");
 IIwritedb(temp_table);
 IIwritedb(".");
 IIwritedb(satt[i].a_name);
 IIwritedb(")))");
 IIsqRinit((char *)0);
 if (IIerrtest()==0) {
   if (IInextget() !=0) {
    IIretdom(1,30,4,&pp);
   }
   IIsqFlush((char *)0);
 }
}

{
 if (IIcsrOpen((char *)0,"cursor_output8","db3",0,media_name) != 0) {
   IIwritedb("retrieve(ISfm5=");
   IIwritedb(media_name);
   IIwritedb(".");
   IIwritedb("f_id,ISdescr1=");
   IIwritedb(media_name);
   IIwritedb(".descrp,");

   IIwritedb("res=");
   IIwritedb(media_name);
   IIwritedb(".");
   IIwritedb("resolution,");


   IIwritedb("sz=");
   IIwritedb(media_name);
   IIwritedb(".");
   IIwritedb("size,");

   IIwritedb("s_rate=");
   IIwritedb(media_name);
   IIwritedb(".");
```

```
    IIwritedb("samp_rate,");

    IIwritedb("enc=");
    IIwritedb(media_name);
    IIwritedb(".");
    IIwritedb("encoding, ");

    IIwritedb("sid=");
    IIwritedb(media_name);
    IIwritedb(".");
    IIwritedb("s_id,");

    IIwritedb("dur=");
    IIwritedb(media_name);
    IIwritedb(".");
    IIwritedb("duration");

    IIwritedb(")");
    IIwritedb(" where ");
    IIwritedb(media_name);
    IIwritedb(".s_id=");
    IIwritedb(sound_value);
    IIcsrQuery ((char *)0);
  }
}
pp=1;
{
    while (look_more==0) {
     if (IIcsrFetch((char *)0, "cursor_output8","db3") != 0) {

       IIcsrRet(1,32,0,ISfn5);
       IIcsrRet(1,32,0,ISdescr1);
       IIcsrRet(1,30,4,&res);
       IIcsrRet(1,30,4,&sz);
       IIcsrRet(1,30,4,&s_rate);
       IIcsrRet(1,30,4,&enc);
       IIcsrRet(1,30,4,&sid);
       IIcsrRet(1,30,4,&dur);

       strcpy(file_name, ISfn5);
       strcpy(snd_record[snd_index].f_id, file_name);
       strcpy(descrp, ISdescr1);
       strcpy(snd_record[snd_index].descrp, descrp);
```

```
        snd_record[snd_index].resolution = res;
        snd_record[snd_index].size = sz;
        snd_record[snd_index].samp_rate = s_rate;
        snd_record[snd_index].encoding = enc;
        snd_record[snd_index].s_id = sid;
        snd_record[snd_index].duration = dur;

        snd_value[snd_index]=snd_record[snd_index].s_id;/*-------*/
        att_array[att_cursor].value_entry=snd_index;

        printf("\n");
        IIcsrEFetch((char *)0);
        qq++;
        if (qq==pp) {
          look_more = 1;
        }
       }
      }
      IIcsrClose((char *)0,"cursor_output8","db3");
    }

  init_buffer(sound_value, 20);
}
/***********************************************
When mode is MODIFY, this function gets the image file atts from the related media
table.
***********************************************/
get_file_id(media_name, i, value)
STR_name media_name;
int i;
int value;
{
  int entry;
 int att_cursor;
  int desired_tupleno;
  int k=0, j=0, look_more=0;
  char ISfn1[FILENAMELEN+1];
  char ISdescr1[DESCRLEN+1];

  char image_value[20];


  int hght = 0;
```

```
int wdth = 0;
int dpth = 0;
int iid = 0;

STR_path f_name;
STR_descrp nothing;
char temp_file[100];  /* Declare more to avoid bus error */
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.   */


inttostr(value, image_value);
{
  IIsqInit ((char *)0);
  IIwritedb("retrieve unique(k=(count(");
  IIwritedb(media_name);
  IIwritedb(".");
  IIwritedb("i_id");
  IIwritedb(")))");
  IIsqRinit((char *)0);
  if (IIerrtest()==0) {
    if (IInextget() !=0) {
     IIretdom(1,30,4,&k);
    }
    IIsqFlush((char *)0);
  }
}

{
  if (IIcsrOpen((char *)0,"cursor_output1","db",0,media_name) != 0) {
    IIwritedb("retrieve(ISfn1=");
    IIwritedb(media_name);
    IIwritedb(".");
    IIwritedb("f_id,ISdescr1=");
    IIwritedb(media_name);
```

```
    IIwritedb(".descrp,");

    IIwritedb("hght=");
    IIwritedb(media_name);
    IIwritedb(".");
    IIwritedb("height,");

    IIwritedb("iid=");
    IIwritedb(media_name);
    IIwritedb(".");
    IIwritedb("i_id,");

    IIwritedb("wdth=");
    IIwritedb(media_name);
    IIwritedb(".");
    IIwritedb("width,");

    IIwritedb("dpth=");
    IIwritedb(media_name);
    IIwritedb(".");
    IIwritedb("depth");

    IIwritedb(")");
    IIwritedb(" where ");
    IIwritedb(media_name);
    IIwritedb(".");
    IIwritedb("i_id");
    IIwritedb("=");
    IIwritedb(image_value);
    IIcsrQuery ((char *)0);
  }
}
k=1;/*-----------------------------*/
{
    while (look_more==0) {
     if (IIcsrFetch((char *)0, "cursor_output1","db") != 0) {
       IIcsrRet(1,32,0,ISfn1);
       IIcsrRet(1,32,0,ISdescr1);

       IIcsrRet(1,30,4,&hght);
       IIcsrRet(1,30,4,&iid);
       IIcsrRet(1,30,4,&wdth);
       IIcsrRet(1,30,4,&dpth);
```

```
        strcpy(f_name, ISfn1);
        strcpy(img_record[img_index].f_id, f_name);
        strcpy(descrp, ISdescr1);
        strcpy(img_record[img_index].descrp, descrp);
        img_record[img_index].height = hght;
        img_record[img_index].i_id = iid;
        img_record[img_index].width = wdth;
        img_record[img_index].depth = dpth;
        /*-----------------------------------*/
        img_value[img_index]=img_record[img_index].i_id;
        att_array[att_cursor].value_entry=img_index;
        /*-----------------------------------*/
        printf("\n");
        IIcsrEFetch((char *)0);
        j++;
        if (j==k) {
          look_more = 1;
        }
      }
     }
     IIcsrClose((char *)0,"cursor_output1","db");
   }
/*  printf("\nimg_record[img_index].i_id =>%d",img_record[img_index].i_id);
  printf("\nimg_record[img_index].f_id =>%s",img_record[img_index].f_id);
  printf("\nimg_record[img_index].descrp =>%s",img_record[img_index].descrp);
  sleep(1);*/
  init_buffer(image_value, 20);
}
/************************************************
When mode is modify, this function helps user modify the tuples in the result table one
by one.
************************************************/
process_tuple_by_tuple(r)
int r;
{
  int entry;
  int desired_tuple;
  char c_temp[60];
  int count=0;
  int j=0,k=0,l=0,temp;
  char char_value[21],a;
  char file_name[20];
```

```
  int img_value, snd_value;
  int  integer_value,media_value,found,media1_value;
  float real_value;
   int i=0,select=0;
  int g=0;
  int d;
  i_value[i_index]=0;
  desired_tuple=r;
  printf("\nTuple to be modified ::  Tuple # %d ", desired_tuple+1);
  sleep(3);
/* # line 3169 "db.sc" */  /* select */
  {
    IIsqInit((char *)0);
    IIwritedb("retrieve(g=(count(");
    IIwritedb(temp_table);
    IIwritedb(".");
    IIwritedb(satt[0].a_name);
    IIwritedb(")))");
    IIsqRinit((char *)0);
    if (IIerrtest() == 0) {
      if (IInextget() != 0) {
        IIretdom(1,30,4,&g);
      } /* IInextget */
      IIsqFlush((char *)0);
    } /* IIerrtest */
  }
l=0;
if (g==0) {
       printf("\nPress ENTER to continue...");
       a=getchar();
       return;
       }
/* # line 3171 "db.sc" */  /* host code */
    if (IIcsrOpen((char *)0,"cursor_output","db2",0,temp_table) != 0) {
      IIwritedb("retrieve(");
      for (select=0;select<n-1;select++) {
      IIwritedb(satt[select].a_name);
      IIwritedb("=");
      IIwritedb(temp_table);
      IIwritedb(".");
      IIwritedb(satt[select].a_name);
      IIwritedb(",");
       }
```

```
    IIwritedb(satt[select].a_name);
    IIwritedb("=");
    IIwritedb(temp_table);
    IIwritedb(".");
    IIwritedb(satt[select].a_name);
    IIwritedb(")");
    IIcsrQuery((char *)0);
  } /* IIcsrOpen */
printf("\n");
look_more=0;
l=0;
if (g==0) {
  look_more=1;
}
table_cursor = table_entry;
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;
i_index=0;
c_index=0;
/* 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","db2") != 0) {
    if (desired_tuple == 1){
     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);
         if (desired_tuple == 1){
           printf("%s :  %s",satt[i].a_name,char_value);
           strcpy(c_temp, char_value);
           strcpy(c_value[c_index], c_temp);
           att_array[att_cursor].value_entry = c_index;
           c_index = (c_index + 1) % 20;
           att_cursor = att_array[att_cursor].next_index;
         }
       }
       if (strcmp(satt[i].data_type,"integer")==0) {
```

```
        IIcsrRet(1,30,4,&integer_value);
        if (desired_tuple == 1){
          printf("%s :  %d  ",satt[i].a_name,integer_value);
          i_value[i_index]=integer_value;
          att_array[att_cursor].value_entry = i_index;
          i_index = (i_index + 1) % 20;
          att_cursor = att_array[att_cursor].next_index;
        }
      }
      if (strcmp(satt[i].data_type,"float")==0) {
        IIcsrRet(1,31,4,&real_value);
        if (desired_tuple == 1){
          printf("%s :  %8.2f  ",satt[i].a_name,real_value);
        }
      }
      if (strcmp(satt[i].data_type,"image")==0) {
        IIcsrRet(1,30,4,&media_value);
        if (desired_tuple == 1){
          img_value=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);
        if (desired_tuple == 1){
          snd_value=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==g) {    /* check if no more data to print */
     look_more =1;  /* exit of the loop */
    }
  } /* IIcsrFetch */
} /* end while */

IIcsrClose((char *)0,"cursor_output","db2"); /* close the cursor */

printf("Press ENTER to continue ..");
a= getchar();
```

```
  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();
      printf("\nThe attribute values will be read from the media table  %s", media_name);
      sleep(2);
      get_file_id(media_name, i, img_value);
      printf("Press ENTER to continue ..");
      a= getchar();
      att_cursor = att_array[att_cursor].next_index;
      media_counter++;
      media_value=0;/*-------------*/
    }
    if (strcmp(satt[i].data_type,"sound")==0) {
      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();
      printf("\nThe attribute values will be read from the media table  %s", media_name);
      sleep(2);
      get_snd_file_atts(media_name, i, snd_value);
      printf("Press ENTER to continue ..");
      a= getchar();
      att_cursor = att_array[att_cursor].next_index;
      media_counter++;
      media1_value=0;/*-------------*/
    }
  } /* end for select < n*/

}
/*****************************************************************
When mode is MODIFY, this function prints the number of tuples in the result table.
*****************************************************************/
int print_for_modify(c)
int c;
{
  c=0;
  /* # line 3169 "db.sc" *//* select */
  {
```

```
    IIsqInit((char *)0);
    IIwritedb("retrieve(c=(count(");
    IIwritedb(temp_table);
    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 */
  }
  printf("\n*** THERE ARE %d RECORDS (TUPLES) TO BE MODIFIED ***",c);
  printf("\n(You will be queried for modifying each tuple)");
  sleep(3);
  return(c);
}
/*********************
When mode is modify, this function deletes the modified tuples from the tables.
*********************/
void delete_formatted_part_for_modify()
{
  int i;
  printf("\nThe tuples that match the delete query are being deleted from table *** %s ***
now",satt[0].t_name);
  printf("\nPress ENTER to continue");
  a=getchar();
  IIsqInit((char *)0);
  IIwritedb("delete ");
  IIwritedb(satt[0].t_name);
  IIwritedb(" where ");
  for (i=0; i<n-1; i++){
    IIwritedb(satt[0].t_name);
    IIwritedb(".");
    IIwritedb(satt[i].a_name);
    IIwritedb("=");
    IIwritedb(temp_table);
    IIwritedb(".");
    IIwritedb(satt[i].a_name);
    IIwritedb(" and ");
  }
```

```
    IIwritedb(satt[0].t_name);
    IIwritedb(".");
    IIwritedb(satt[i].a_name);
    IIwritedb("=");
    IIwritedb(temp_table);
    IIwritedb(".");
    IIwritedb(satt[i].a_name);

  IIwritedb(" ");
  IIsqSync(1,(char *)0);
}


/*************************************************************************
This function, when mode is DELETE, deletes the modified tuples from the related media
tables.
*************************************************************************/
get_rid_image(imageno)
int imageno;
{
  IIsqInit((char *)0);
  IIwritedb("delete ");
  IIwritedb(media_name);
  IIwritedb(" where ");
  IIwritedb(media_name);
  IIwritedb(".");
  IIwritedb("i_id =");
  IIwritedb(temp_table);
  IIwritedb(".");
  IIwritedb(satt[imageno].a_name);
  IIwritedb(" ");
  IIsqSync(1,(char *)0);
}
/*************************************************************************
This function, when mode is MODIFY, deletes the modified tuples from the related media
tables.
*************************************************************************/
mod_get_rid_image(imageno, value)
int imageno;
int value;

{
  char media_value[20];
```

```
  inttostr(value, media_value);
  {
    IIsqInit((char *)0);
    IIwritedb("delete ");
    IIwritedb(media_name);
    IIwritedb(" where ");
    IIwritedb(media_name);
    IIwritedb(".");
    IIwritedb("i_id");
    IIwritedb("=");
    IIwritedb(media_value);
    IIwritedb(" ");
    IIsqSync(1,(char *)0);
  }
}
/*************************************************************************
This function, when mode is DELETE, deletes the modified tuples from the related media
tables.
*************************************************************************/
get_rid_sound(soundno)
int soundno;
{

  {
    IIsqInit((char *)0);
    IIwritedb("delete ");
    IIwritedb(media_name);
    IIwritedb(" where ");
    IIwritedb(media_name);
    IIwritedb(".");
    IIwritedb("s_id =");
    IIwritedb(temp_table);
    IIwritedb(".");
    IIwritedb(satt[soundno].a_name);
    IIwritedb(" ");
    IIsqSync(1,(char *)0);

  }
}
/************************************
This function, when mode is MODIFY, deletes the modified tuples from the related media
tables.
************************************/
```

```
mod_get_rid_sound(soundno, value)
int soundno;
int value;
{
  char sound_value[20];
  inttostr(value, sound_value);
  {
    IIsqInit((char *)0);
    IIwritedb("delete ");
    IIwritedb(media_name);
    IIwritedb(" where ");
    IIwritedb(media_name);
    IIwritedb(".");
    IIwritedb("s_id");
    IIwritedb("=");
    IIwritedb(sound_value);
    IIwritedb(" ");
    IIsqSync(1,(char *)0);
  }
}
/*******************************************
When mode is DELETE, this functions is the main function calling other functions to
delete the tuples from the related media tables.
*******************************************/
void ql_print_delete_data()
{
   int 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;
   int c=0;
   /* # line 3169 "db.sc" */       /* select */
   {
        IIsqInit((char *)0);
       IIwritedb("retrieve unique(c=(count(");
        IIwritedb(temp_table);
        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 DELETE query",c);
   if (c==0) {
            printf("\nPress ENTER to continue...");
                  a=getchar();
                  return;
          }
 /* # line 3171 "db.sc" */     /* host code */
    if (IIcsrOpen((char *)0,"cursor_output","db1",0,temp_table) != 0) {
     IIwritedb("retrieve (");
     for (select=0;select<n-1;select++) {
       IIwritedb(satt[select].a_name);
       IIwritedb("=");
       IIwritedb(temp_table);
       IIwritedb(".");
       IIwritedb(satt[select].a_name);
       IIwritedb(",");
     }
     IIwritedb(satt[select].a_name);
     IIwritedb("=");
     IIwritedb(temp_table);
     IIwritedb(".");
     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 temp_tablerelation which is the intermediate table
      hold the temp_tablefrom 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 ENTER to continue ..");
       /* stop before change to the next function so
           the user can see the temp_tableon screen, until he hit ENTER 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) {
      if (image_flag==TRUE){
        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();
        printf("\nmedia_name--> ***%s***", media_name);
        sleep(2);
        get_rid_image(i);
       }
      }
      if (strcmp(satt[i].data_type,"sound")==0) {
       if (sound_flag==TRUE){
        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();
        printf("\nmedia_name--> ***%s***", media_name);
        sleep(2);
        get_rid_sound(i);
       }
      }
    } /* end for select < n*/
      printf("\n");
 }
/*****************************************************************************
When mode is MODIFY, this function checks the media desription if the media data is
modifed.
*****************************************************************************/
int mod_chk_description(file_id, descrp, err_message)
STR_path *file_id;
STR_descrp *descrp;
char *err_message;
{
  int i=0;
  int error = FALSE;
  while (i<1 && !error){
    *err_message = '\0';
    if (strcmp(descrp, " ") != 0)
      error = connect_parser(file_id, descrp, err_message);
    i++;
  }

   if (error)
```

```
    {
      printf("\nThe description for media is NOT acceptable!");

      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 occurred in prolog!\n");
      printf("\nPlease modify it. Thank you!");
      putchar('\007');
      while((c=getchar()) != '\n')
      ;
      return(TRUE);
    }
  else
    return(FALSE);
}
/***********************************
Gets all atts of a given table and puts them in satt array for retrieving all the attributes
of that table.
***********************************/
void get_all_atts_of_a_given_table()
{
  int i = 0,
      count = 0;
  count = table_array[table_list[table_cursor]].att_count;
  n = count;
  att_cursor = table_array[table_list[table_cursor]].att_entry;
  for (i = 0; i < count; i++) /*  Loop to get value for each attribute */
    {
      strcpy(satt[i].t_name, stab[0].t_name);
      strcpy(satt[i].a_name, att_array[att_cursor].att_name);
      strcpy(satt[i].data_type, att_array[att_cursor].data_type);
      att_cursor = att_array[att_cursor].next_index;
    }  /* End of for loop */
}  /* End of get_tuple_value */

/*****************************************************************************
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
This function also handles DELETE and MODIFY operations
/*************************************************************************
void retrieve(mode)
{
  int entry;
  int count;
  int h,r,flag=TRUE;
  int o, u;
  char buf0[13];
  char buf1[13];
  char buf2[13];
  char buf3[13];
  char buf4[13];
  char temp[3];
  char aggregate0[3];
  char aggregate1[3];
  char aggregate2[3];
  char aggregate3[3];
  char aggregate4[3];
  int i,j,x,y,z,found=0;
  int level_no=0, counter=1;
  char table_name[20],attname[20],att_type[20],Ans,More,a;
  char choice;
  init_buffer(buff,100);
  init_buffer(temp_table1,20);
  init_buffer(temp_table2,20);
  init_buffer(temp_table,20);
  choice='0';
  m=0;
  i=0;
  k=0;
  gcond=0;
  numcon=0;
  aggregate_found=FALSE;
  more_selections=TRUE;
  more_levels=TRUE;

  init();
  drop_temp_media_tables();

  while (more_levels != FALSE){
    while (more_selections != FALSE){
```

```
      init_buffer(buff,100);
      printf("\nEnter table name to hold the temporary result of the query: ");
      gets(buff);
      strcpy(temp_table, buff);
      init_buffer(buff,100);

      help_tables(buff);
      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) {
            stab[i].t_name[j]= '\0';
            j=55;
            k=k+1;
            i=i+1;
           }
          else {
            if (buff[k] == ' ')
            j=55;  /* Skip the white space if the user typped in*/
            else
            stab[i].t_name[j]=buff[k];
            if (buff[k]==0) { /* if null value in buffer (end of string) */
            m=i+1;
            j=55;
            i=1000;
            }

            k=k+1;
            }
         }
      }/*end while*/

      strcpy(temp_table1, stab[0].t_name);

      for (i=0;i<m;i++) {
       strcpy(table_array[table_index].table_name, stab[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 !!!" ,stab[i].t_name);
          printf("\nPress ENTER to continue !!");
          a=getchar();
```

```
        return;
      } /* end else */
    } /* end for loop */

     /* Specify the join condition if there are more than 2 table select */
    help_join();

     /* Select attribute */
    init_buffer(buff,100);
    i = 0;
    j = 0;
    k = 0;
    x = 0;
    z = 0;
    if (mode == RTRVE_MODE){
     /* 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) separated by comma <,> - <?> for HELP ! -\nhit
<ESC> for no attribute");
         printf("\nSELECT ATTRIBUTE(S) : ");
         gets(buff);
         if (strcmp(buff,"?")==0) {
           p_att(stab[y].t_name);
         } /* end if buff == "?"  */
       } /* end while need help */

       while (i < 100) {
         for (j=0;j<13;j++){
           if (buff[k]==27) {
           goto start_again;
           }
           if (buff[k]==44) {
           buf0[j]= '\0';
           strcpy(satt[x].t_name, stab[y].t_name);
           init_buffer(temp,3);
           init_buffer(aggregate0,3);
           u=x;
           aggregate_found=FALSE;
           aggregate_found=check_aggregate(buf0, temp, aggregate_found);
           printf("\n");
```

```
strcpy(aggregate0, temp);
if (aggregate_found==TRUE){
  get_attribute(buf0, satt[u].a_name);
  printf("\n");
  if (strcmp(aggregate0,"cnt")==0)
    satt[u].aggregate_type=1;
  if (strcmp(aggregate0,"sum")==0)
    satt[u].aggregate_type=2;
  if (strcmp(aggregate0,"avg")==0)
    satt[u].aggregate_type=3;
  if (strcmp(aggregate0,"max")==0)
    satt[u].aggregate_type=4;
  if (strcmp(aggregate0,"min")==0)
    satt[u].aggregate_type=5;
  printf("\n");
}
if (aggregate_found==FALSE){
  strcpy(satt[u].a_name,buf0);
  satt[u].aggregate_type=0;
  printf("\n");
  clr_scr();
}
j=55;
k=k+1;
i=i+1;
x=x+1;
}
else {
if (buff[k] == ' ')
  j=55;  /* Skip the white space if user typped in */
else{
  buf0[j]=buff[k];
}
if (buff[k]==0) {
  strcpy(satt[x].t_name, stab[y].t_name);
  init_buffer(temp,3);
  init_buffer(aggregate0,3);
  u=x;
  aggregate_found=FALSE;
  aggregate_found=check_aggregate(buf0, temp, aggregate_found);
  printf("\n");
  strcpy(aggregate0, temp);
  if (aggregate_found==TRUE){
```

```
          get_attribute(buf0, satt[u].a_name);
          printf("\n");
          if (strcmp(aggregate0,"cnt")==0)
            satt[u].aggregate_type=1;
          if (strcmp(aggregate0,"sum")==0)
            satt[u].aggregate_type=2;
          if (strcmp(aggregate0,"avg")==0)
            satt[u].aggregate_type=3;
          if (strcmp(aggregate0,"max")==0)
            satt[u].aggregate_type=4;
          if (strcmp(aggregate0,"min")==0)
            satt[u].aggregate_type=5;
          printf("\n");
         }
         if (aggregate_found==FALSE){
           strcpy(satt[u].a_name,buf0);
           satt[u].aggregate_type=0;
           printf("\n");
           clr_scr();
         }

         n=x+1;
         j=55;
         i=1000;
         }
       k=k+1;
       } /* end else */
     } /* end for j < 13 */
   }/*end while */
   x=x+1;
start_again:
   k=0;
   init_buffer(buff,100);
   i=0;
} /* End select attribute for each table go to the next table */

clr_scr();
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);
}
}  /*  closure of if mod ==ret  */
if ((mode==DEL_MODE) || (mode==MOD_MODE)){
```

```
 table_cursor = table_entry;
 get_all_atts_of_a_given_table();
}

printf("\n");
cond=0;
printf("\nAny condition ? (y/n) :");
Ans=yes_no_answer();
if ((Ans==121)||(Ans==89)){
 choice=nested_processcondition(choice,temp_table1,temp_table2,temp_table);
}

if (choice=='0'){
 ql_retrieve(temp_table);
 ql_printdata(temp_table);
}
init_buffer(buff,100);

query_for_intersect_union(choice,temp_table1,temp_table2,temp_table);
printf("\nMore selections at this level ? (y/n)");
Ans=yes_no_answer();
if ((Ans==121)||(Ans==89)){
 more_selections=TRUE;
 choice='0';
 y = 0;
 j = 0;
 x = 0;
 z = 0;
 m=0;
 i=0;
 k=0;
 cond=0;
 gcond=0;
 numcon=0;
 n=0;
 found=0;
 init();
 drop_temp_media_tables();
 init_buffer(buff,100);
 init_buffer(temp_table1,20);
 init_buffer(temp_table2,20);
 init_buffer(temp_table,20);
}
```

```
else{
 more_selections=FALSE;
 printf("\n");
}
}
printf("\nMore levels ? (y/n)");
 Ans=yes_no_answer();
 if ((Ans==121)||(Ans==89)){
   more_levels=TRUE;
   more_selections=TRUE;
   level_no=level_no+1;
   choice='0';
   y = 0;
   j = 0;
   x = 0;
   z = 0;
   m=0;
   i=0;
   k=0;
   cond=0;
   gcond=0;
   numcon=0;
   n=0;
   init();
   drop_temp_media_tables();
   init_buffer(buff,100);
   init_buffer(temp_table1,20);
   init_buffer(temp_table2,20);
   init_buffer(temp_table,20);
   found=0;
 }
 else{
   more_selections=FALSE;
   more_levels=FALSE;
 }
}/* end while more levels */

if (mode==DEL_MODE){
 image_flag=TRUE;
 sound_flag=TRUE;
 printf("\nDo want to continue with DELETION ? (y/n) ::");
 Ans=yes_no_answer();
 if ((Ans==110)||(Ans==78))
```

```
        goto qquit;
      ql_print_delete_data();
      delete_formatted_part_for_modify();
      drop_table(temp_table);
      image_flag = FALSE;
      sound_flag = FALSE;
     }
     if (mode==MOD_MODE){
      formatted_flag = FALSE;
      image_flag = FALSE;
      sound_flag = FALSE;
      h=print_for_modify(h);
      for (r=0; r<h; r++){
        formatted_flag = FALSE;
        image_flag = FALSE;
        sound_flag = FALSE;
        media_counter = 0;
        process_tuple_by_tuple(r);
        mod_display_tuple(mode, media_counter);
        store_data(mode);
        mod_ql_insert_tuple(mode);
        att_cursor = 0;/*to initialize the value arrays */
        img_index = 0;
        snd_index = 0;
        i_index = 0;
        f_index = 0;
        c_index = 0;
        delete_for_modify(r);
      }
      delete_formatted_part_for_modify();
      drop_table(temp_table);
      image_flag = FALSE;
      sound_flag = FALSE;
     }
   qquit:
     printf("\n ");
} /* End procedure */
```

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 52 Naval Postgraduate School Monterey, California 93943-5100	2
3.	Center for Naval Analysis 4401 Ford Ave. Alexandria, Virginia 22302-0268	1
4.	John Maynard Code 042 Command and Control Departments Naval Ocean Systems Center San Diego, California 92152	1
5.	Dr. Sherman Gee ONT-221 Chief of Naval Research 880 N. Quincy Street Arlington, Virginia 22217-5000	1
6.	Leah Wong Code 443 Command and Control Departments Naval Ocean Systems Center San Diego, California 92152	1
7.	Professor Vincent Y. Lum Code CsLm Department of Computer Science Naval Postgraduate School Monterey, California 93943	1

8. Deniz Kuvvetleri Komutanligi — 1
Personel Daire Başkanligi
Bakanliklar, Ankara / TURKEY

9. Deniz Harp Okulu Komutanligi — 1
81704 Tuzla, Istanbul / TURKEY

10. Professor C. Thomas Wu — 1
Code CsWq
Department of Computer Science
Naval Postgraduate School
Monterey, California 93943

11. Professor Klaus Meyer-Wegener — 1
University of Erlangen-Nuernberg
IMMD VI, Martensstr.3,
8250 Erlangen / GERMANY

12. Dr. Bernhard Holtkamp — 1
University of Dortmund
Department of Computer Science
Software Technology
P.O. Box 500
D-4600 Dortmund 50 / GERMANY

13. Huseyin Aygun — 1
Gultepe Mahallesi
Ture Sokak, 40
Sefakoy
Istanbul / TURKEY