



Calhoun: The NPS Institutional Archive
DSpace Repository

Theses and Dissertations

1. Thesis and Dissertation Collection, all items

1996-06

Reactivation of the relational interface in
M_x001B_p2_x001B_sDBMS and
implementation of the EWIR database

Scrivener, Donna N.; Edwards, Renell D.

Monterey, California. Naval Postgraduate School

<https://hdl.handle.net/10945/32201>

Downloaded from NPS Archive: Calhoun



Calhoun is the Naval Postgraduate School's public access digital repository for research materials and institutional publications created by the NPS community. Calhoun is named for Professor of Mathematics Guy K. Calhoun, NPS's first appointed -- and published -- scholarly author.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

<http://www.nps.edu/library>

NAVAL POSTGRADUATE SCHOOL MONTEREY, CALIFORNIA



THESIS

**REACTIVATION OF THE RELATIONAL
INTERFACE IN M²DBMS AND IMPLEMENTATION
OF THE EWIR DATABASE**

by

Donna N. Scrivener
and
Renell D. Edwards

June 1996

Thesis Advisor:

C. Thomas Wu

Approved for public release; distribution is unlimited.

DTIC QUALITY INSPECTED 3

19960910 144

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including time reviewing instructions, searching existing data sources gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Washington Headquarters Services, Directorate for Information, Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204 Arlington VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (07045-0188), Washington, DC 20503				
1. Agency use only (leave Blank)	2. REPORT DATE June 1996		3. REPORT TYPO AND DATES COVERED Master's Thesis	
4. TITLE AND SUBTITLE REACTIVATION OF THE RELATIONAL INTERFACE IN M²DBMS AND IMPLEMENTATION OF THE EWIR DATABASE			5. FUNDING NUMBERS	
6. AUTHOR(S) Donna N. Scrivener, Renell D. Edwards				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)			10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U. S. Government.				
12A. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			12B. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 Words) The primary Department of Defense source for technical parametric performance data on non-communications emitters is the Electronic Warfare Reprogramming Database (EWIRDB). Data representation in the EWIRDB is via disjointed parametric tree models which are implementation oriented. These parametric trees obscure the intended semantics and representation of the data, making the database difficult to use and understand. The problem addressed by this thesis is to determine if the relational model and the relational interface of the Multimodel and Multilingual Database System (M ² DBMS) in the Laboratory for Database Systems Research at the Naval Postgraduate School is capable of supporting a representative subset of the EWIRDB. We implemented a representative portion of the EWIR database on the relational interface of the M ² DBMS. In order to accomplish this the relational interface was reactivated and returned to its original operational state and fully tested to determine its capabilities. In addition, the schema and an instance of a relational EWIR data model must be developed for implementation. The relational interface was successfully returned to its original operational state. Significant limitations in the interface's ability to process queries were discovered, however, in that the system can not query schema of greater than four relations.				
14. SUBJECT TERMS M2 DBMS, Relational Interface, EWIR, Multimodel Database			15. NUMBER OF PAGES 132	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL	

NSN 7540-01-280-5500

STANDARD FORM 298 (REV. 2-89)
PRESCRIBED BY ANSI STD. Z39-18 298-102**DTIC QUALITY INSPECTED 3**

SECRET

Approved for public release; distribution is unlimited

**REACTIVATION OF THE RELATIONAL INTERFACE IN M²DBMS AND
IMPLEMENTATION OF THE EWIR DATABASE**

Donna N. Scrivener
Lieutenant Commander, United States Navy
B.A., State University of New York, 1981

Renell D. Edwards
Lieutenant, United States Navy
B.S., Florida Agricultural and Mechanical University, 1988

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL

June 1996


Authors:


Donna N. Scrivener


Renell D. Edwards

Approved by:


C. Thomas Wu, Thesis Advisor


David K. Hsiao, Second Reader


Ted Lewis, Chairman
Department of Computer Science

ABSTRACT

The primary Department of Defense source for technical parametric performance data on non-communications emitters is the Electronic Warfare Reprogramming Database (EWIRDB). Data representation in the EWIRDB is via disjointed parametric tree models which are implementation oriented. These parametric trees obscure the intended semantics and representation of the data, making the database difficult to use and understand. The problem addressed by this thesis is to determine if the relational model and the relational interface of the Multimodel and Multilingual Database System (M²DBMS) in the Laboratory for Database Systems Research at the Naval Postgraduate School is capable of supporting a representative subset of the EWIRDB.

We implemented a representative portion of the EWIR database on the relational interface of the M²DBMS. In order to accomplish this the relational interface was reactivated and returned to its original operational state and fully tested to determine its capabilities. In addition, the schema and an instance of a relational EWIR data model must be developed for implementation.

The relational interface was successfully returned to its original operational state. Significant limitations in the interface's ability to process queries were discovered, however, in that the system can not query schema of greater than four relations.

TABLE OF CONTENTS

I.	INTRODUCTION	1
	A. AN OVERVIEW OF THE M2DBMS	1
	B. RESEARCH GOALS	3
II.	THE EWIR DATABASE	5
	A. WHY THE EWIR DATABASE	5
	B. TRANSLATION FROM THE OBJECT-ORIENTED MODEL TO THE RELATIONAL MODEL.	6
III.	ACTIVATION OF THE RELATIONAL INTERFACE	15
	A. STARTING UP THE INTERFACE	15
	B. TROUBLESHOOTING	18
	C. A SUCCESSFUL RUN	18
IV.	IMPLEMENTATION OF THE EWIR DATABASE	31
	A. THE EWIRsql db FILE	31
	B. MASS LOADING THE DATA	32
V.	SQL TRANSACTIONS	39
	A. QUERYING THE COURSE DATABASE	39
	B. QUERIES OF THE EWIR DATABASE	41
	C. QUERY LIMITATIONS OF THE RELATIONAL INTERFACE	42
	D. THE EWIR DATABASE QUERIES	43
	E. SQL QUERY LIMITATIONS IN M2DBMS	50
VI.	CONCLUSION	61
	A. SUMMARY	61

B. SUITABILITY OF THE RELATIONAL INTERFACE FOR EWIR	61
C. FUTURE RESEARCH SUGGESTIONS	62
LIST OF REFERENCES	63
APPENDIX A COURSE SAMPLE DATABASE RUN SCRIPT	65
APPENDIX B EWIR DATABASE SUCCESSFUL LOAD SCRIPT	89
INITIAL DISTRIBUTION LIST	123

I. INTRODUCTION

Research in the development of the Multimodel and Multilingual Database System (M²DBMS) has been ongoing for over a decade. This thesis specifically builds on a recently completed two-thesis project, both of which were involved with the application of object-oriented database management for the Electronic Warfare Integrated Reprogramming (EWIR) data. The first of these two theses [Ref.1] is a conceptual design of the EWIR's existing hierarchical and flat file data collections into an object-oriented database specification. The second [Ref.2] actually implements a subset of the object-oriented EWIR database on the M²DBMS and uses the Object-Oriented Data Definition Language (O-ODDL) to specify and create the database while the Object-Oriented Data Manipulation Language (O-ODML) is used to develop and write a series of object-oriented queries on the newly created database.

A. AN OVERVIEW OF THE M²DBMS

The M²DBMS currently implemented at the Naval Postgraduate School Laboratory for Database Systems Research was conceived in response to the limitations posed by conventional monomodel and monolingual database systems. Traditionally, database systems support a single model and its corresponding language and it is incumbent upon the user to select the model that best meets the database needs, relational vs. object oriented for instance. This homogeneity of database systems can force an organization to operate several different homogenous systems to support its operations.

The M²DBMS is a multi-database system which supports five traditional data models and languages as well its own

kernel data model and language. The conventional data models/data languages currently supported are: relational/SQL, network/CODASYL, hierarchical/DL/1, functional/DAPLEX, and the newly installed object-oriented/O-ODML. M²DBMS differs in that its kernel is based on a common attribute-based data model (ABDM) and its corresponding attribute-based language (ABDL). This kernel uses the attribute-value pair as the basic data unit. This attribute-value pair consists of object identifier, called the attribute, and its associated value. The distinction between "kernelized" databases within the M²DBMS is made by their corresponding schema. It is this process of mapping the user's data model to the attribute-based kernel which distinguishes M²DBMS. The ABDM supports the five fundamental database operations: RETRIEVE, RETRIEVE COMMON, INSERT, UPDATE, and DELETE.

Each database developed on the M²DBMS requires a schema file which outlines the schema of the user's desired database in its respective query language. Loading this file is what allows the language interface layer to generate the descriptor(.d) and template(.t) files. Request files contain transactions that the user wishes to process against the desired database. These transactions must be written in a data language consistent with the data model of the database. The detailed requirements and restrictions of these files are discussed in the M²DBMS User's Manual[Ref.3].

The strength of the M²DBMS lies in its cross-model access: ideally the kernel system should allow translation to occur between heterogenous databases in the system. That is, a user familiar with the relational model can access information from an object-oriented database using a relational query language. Figure 1 shows conceptually how this data sharing occurs.

The advantage of this type of multimodel and

multilingual database system to industry should be obvious. It would allow an organization to choose the best model for their data, relational for personnel data, hierarchical for inventory, etc, and then have the data be widely available without any additional training required in order to retrieve it. It is also the most viable method of integrating legacy databases into a unified, enterprise database.

B. RESEARCH GOALS

With the exception of the object-oriented interface, which was implemented in Reference 2, all of the other interfaces are fairly mature. The problem lay in the fact that they had not been used in several years and were adversely affected by changes made to the system in the intervening years. The goal of this and a sister thesis [Ref. 4] is to reactivate two of these interfaces and then implement the same portion of the subset of the EWIR database implemented in Reference 2. Specifically, we are reactivating the relational interface while Reference 4 reactivates the network interface.

This primary value of this work is to serve as a foundation for the next step in this ongoing project: implementation of the cross-model access, or interoperability. It is anticipated that the first cross-model access to be completed will be between the relational and object-oriented interfaces. The existence of identical sample EWIR databases based on different models and the ability to process the same query set is seen as a prerequisite to further progress in this area.

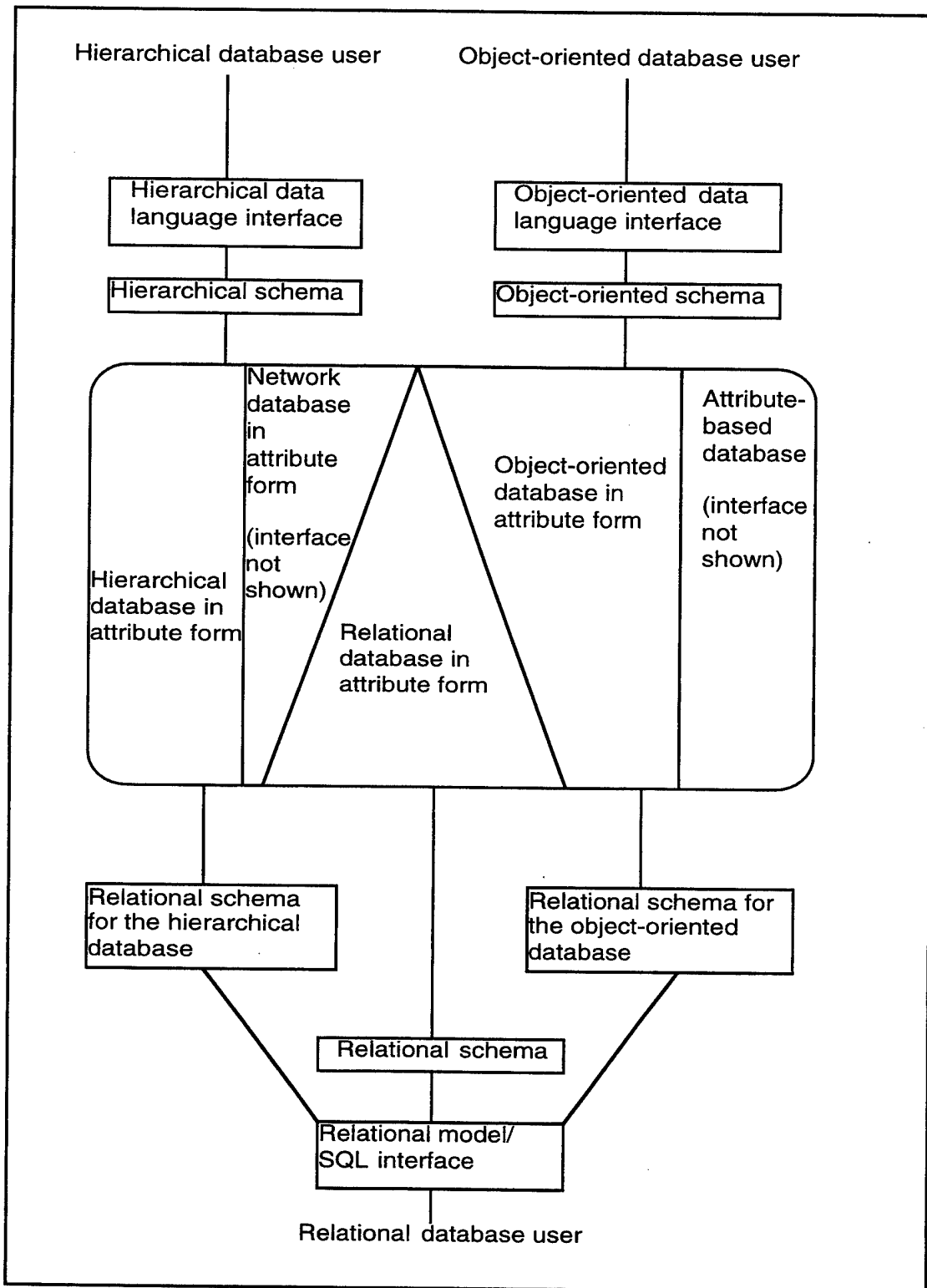


Figure 1 Data Sharing in M²DBMS

II. THE EWIR DATABASE

The development of the proposed object-oriented model [Ref. 1] for the EWIR database and its subsequent test implementation in the M²DBMS [Ref.2] resolved many of the problems inherent in the original EWIR database design. The intent in developing a relational model was not to propose it as a "better" design than the object-oriented model, only to provide a common database and query set as a basis for development of the cross-model access in the M²DBMS. This chapter will provide a brief discussion of the original EWIR database, an overview of the object-oriented design, the translation of a representative portion of the object model to a relational model, and a very brief discussion of the shortcomings of the relational design.

A. WHY THE EWIR DATABASE

The EWIR database is the Department of Defense approved source for technical parametric and performance data on non-communications emitters and associated systems. The EWIR system provides an up-to-date and accurate source of information for reprogramming United States Electronic Warfare (EW) Combat Systems. It contains parametric data on radars, jammers, navigational aids, and numerous non-communication electronic emitters.

The EWIR database was selected as the test database for References 1 and 2 because it is a widely used, real world application which was complex enough to evaluate the utility and versatility of the object-oriented interface. It was also selected because it has been identified by the National Air Intelligence Center (NAIC) as being difficult to understand and use in its present form. The database's easily identifiable objects and relationships between objects

also make it an acceptable candidate for the object-oriented design.

References 1 and 2 report that users of the "old" EWIR database clearly feel that the current hierarchical model depends too heavily on the users' understanding of the data relationships, many of which are not explicitly described in the data model. The deficiencies in capturing data relationship information are also noted. Additionally, previous researchers have found that the EWIR database format is difficult to interpret where codes are not standardized for all record types.

Implementation of a representative subset of the EWIR database as an object-oriented version rectifies most of these shortcomings. Data relationships are imbedded into the data model, the responsibility for knowledge of data relationships is no longer incumbent upon the user. Reference 2 concludes that the object-oriented model results in a more intuitive, natural, and powerful database system.

B. TRANSLATION FROM THE OBJECT-ORIENTED MODEL TO THE RELATIONAL MODEL

The first step in implementing the EWIR database on both the relational and network interfaces in M²DBMS was selection of a representative subset of the database implemented in Reference 2. We then translate the subset into a relational model while Reference 4, our sister thesis, develops a network model for the same subset. Specifically, we agreed that focusing exclusively on the antenna data section of Figure 2 would allow a sufficiently large database for our research needs.

Both Reference 4 and our thesis use the same entity-relationship (ER) diagram, developed cooperatively, as the basis for data model development. This diagram describes

data as entities, relationships and attributes and is shown in Figure 3. Once the ER diagram is complete, mapping to the relational schema is accomplished using the algorithm described by Elmasri and Navathe [Ref.5] and is straightforward. Initially, for each regular entity type from Figure 3 (EMITTER, ANTENNA, RADIATION PATTERN, POLARIZATION, SCAN, TRACK) a relation is created and all simple attributes included. The inheritance specialization, a strength of the object-oriented model, introduces some enhanced-ER modeling concepts such as superclass/subclass relationships which alters the initially developed schema. Thus the disjoint subclass DIRECTIONAL and its attributes are combined with the relation RADIATION PATTERN to form a new relation. Similarly, CIRC_OR_ELLIP POLARIZATION is joined with POLARIZATION. (The treatment of the overlapping subclass MECHANICAL_TARGET_TRACKING and the relation TRACK follows the same pattern.) Multi-level specialization is described by the MECHANICAL SCAN and SECTOR subclasses of the SCAN relation and results in the three entities forming two relations. The schema which ultimately results has seven entities and is shown in Figure 4. (The naming of the relations was constrained by peculiarities of the M²DBMS which are discussed in the relational interface section.)

Less straightforward than the development of the relational schema was the interpretation of the actual data used in Reference 2. As stated previously, there is obvious value in having identical databases to serve as the foundation for implementation of the cross-model access functionality. The difficulty was two-fold. The first lay in the fact that, in order to maintain the unclassified nature of the thesis, the specific data utilized in Reference 2 was gibberish. Secondly, and more significantly, the structure of the object-oriented model and its use of inheritance and separate classes for data types had no direct

application to the relational model. To "fill in the tables" generated by our schema it was necessary to reverse-engineer the data from the EWIR record file from Reference 2 using the sample EWIR record template provided by the authors. The results of that process are presented in Figure 5. This figure is illustrative only, this is not the form used by the M²DBMS. Those details are provided in the implementation section of this thesis.

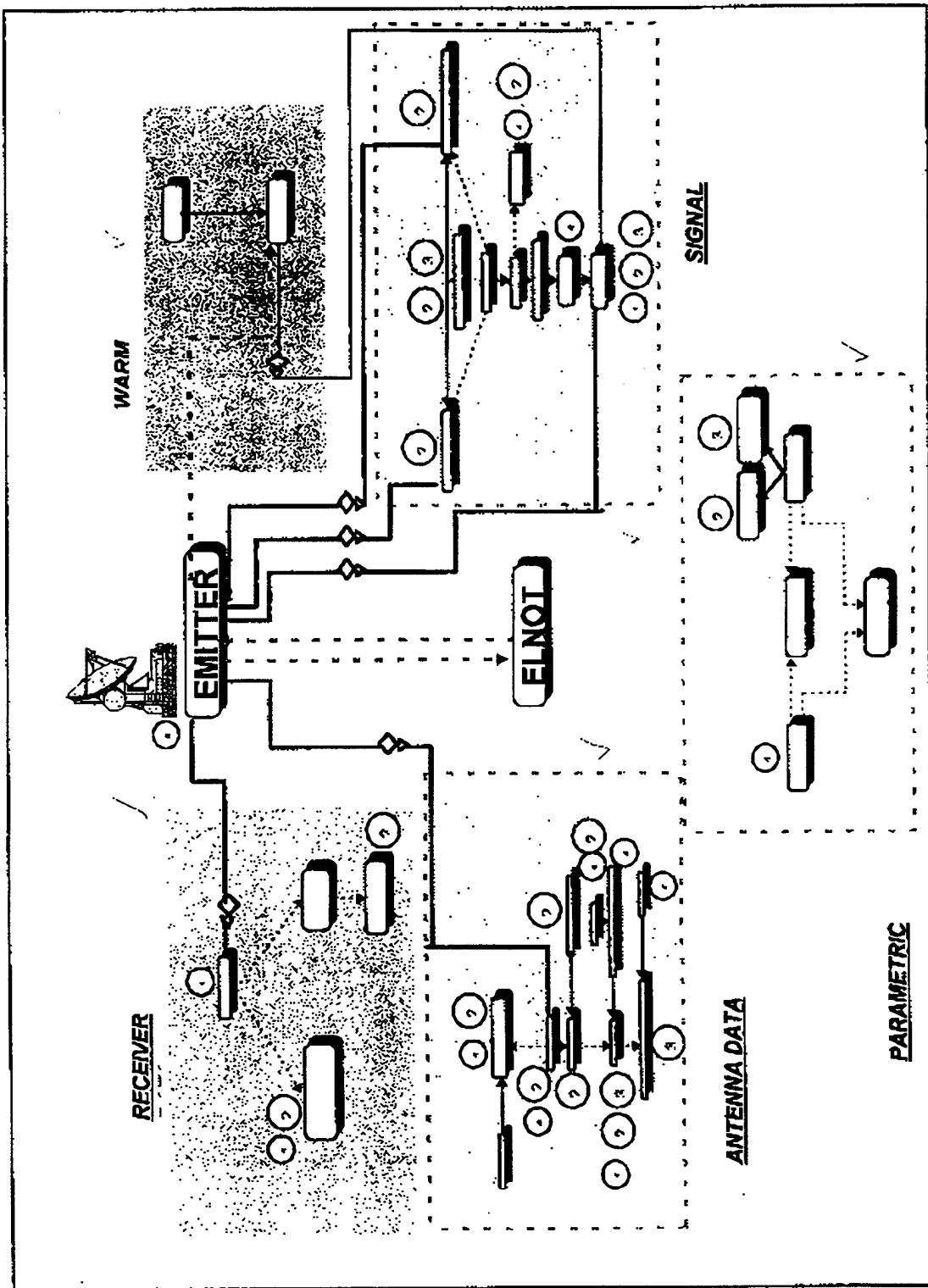


Figure 2 The Top Level View of the New Object-Oriented EWIR Database

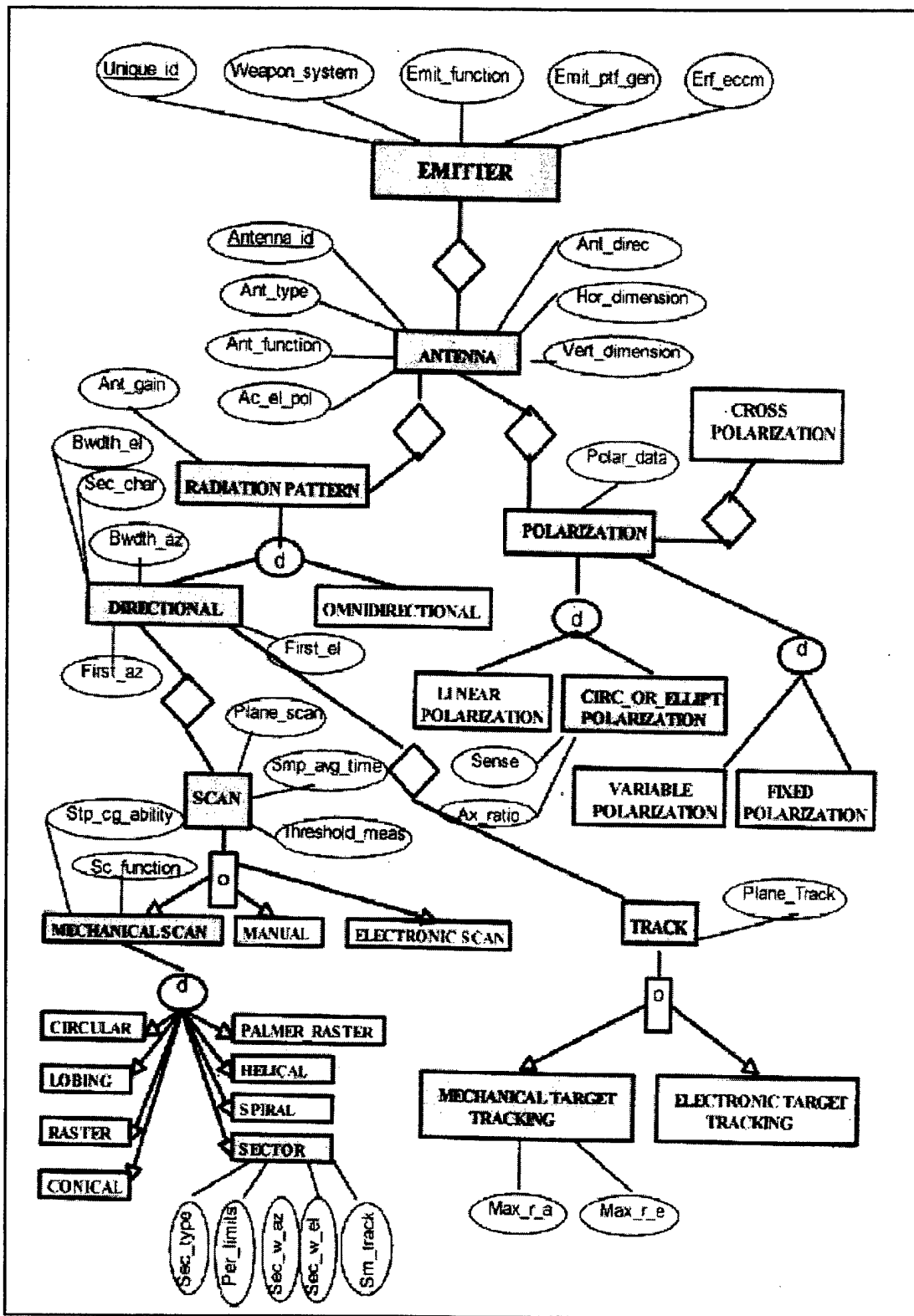


Figure 3 Entity-Relationship Diagram

EMITTER

UNIQUEID	WEAPONSYSTEM	EMITFUNCTION	EMITPTFGEN	ERFECCM
----------	--------------	--------------	------------	---------

ANTENNA

ANTENNAID	ANTTYPE	ANTFUNCT	HORDIM	VERTDIM	ANTDIREC	ACELPOL	UNIQUEID
-----------	---------	----------	--------	---------	----------	---------	----------

DIRECTION

RADPATID	ANTGAIN	SECCHAR	BWDTHAZ	BWDTHEL	FIRSTAZ	FIRSTEL
----------	---------	---------	---------	---------	---------	---------

CIRCORELIP

POLARID	POLARDATA	SENSE	AXRATIO	COMMENT	CONIAGENCY	LASTUPDATE
---------	-----------	-------	---------	---------	------------	------------

MEHSCAN

SCANID	SMPAVGTIME	THRESHOLDMEAS	PLANSKAN	STPSGABILITY	SCFUNCT	RADPATID
--------	------------	---------------	----------	--------------	---------	----------

MEHTRACK

TRACKID	PLANTRACK	UMAXRA	LMAXRA	UMAXRE	LMAXRE
---------	-----------	--------	--------	--------	--------

SECTOR

SCANID	SECTORTYPE	UPPERLIMITS	LOWERLIMITS	SECWAZ	SECWEL	SMTRACK
--------	------------	-------------	-------------	--------	--------	---------

Figure 4 The EWIR Relational Database Schema

EMITTER

UNIQUEID	WEAPONSYSTEM	EMITFUNCTION	EMITPTFGEN	ERFECCM
Ee1	Aa10	Parabolic	Modpulsewave	Ww1
Ee2	Aa6	Modpulsewave	Phasedarray	Ww2
Ee3	Sa21	Phasedarray	Parabolic	Ww3
Ee4	Aa9	Goofy	Minnie	Ww1

ANTENNA

ANTENNAID	ANTTYPE	ANTFUNCT	HORDIM	VERTDIM	ANTDIREC	ACELPOL	UNIQUEID
A1	Phasedarray	Longrngaa	3ft	4ft	Rad1	P1	Ee1
A2	Squaresail	Longrngaa	3ft	4ft	Rad2	P2	Ee2
A3	Parabolic	Longrngaa	325ms	300kw	Rad2	P3	Ee3

MECHSCAN

SCANID	SMPAVGTIME	THRESHOLDMEAS	PLANSCAN	STPSGABILITY	SCFUNCT	RADPATID
Sca1	325ms	4ft	Phasedarray	Phasedarray	Parabolic	Rad1
Sca2	300kw	325ms	Parabolic	Parabolic	Modpulsewave	Rad2
Sca3	300ms	325kw	Parabolic	Parabolic	Modpulsewave	Rad2

Figure 5 Relational Database Instance of EWIR Schema

CIRCORELIP

POLARID	POLARDATA	SENSE	AXRATIO	COMMENT	CONTAGENCY	LASTUPDATE
Pp1	Cross	Left	20db	Cm1	Airforce	8_95
Pp2	Pdata2	Parabolic	300kw	Satellite_det	Airforce	8_95
Pp3	Pdata3	Parabolic	20db	Satellite_det	Airforce	8_95

MECHTRACK

TRACKID	PLANTRACK	UMAXRA	LMAXRA	UMAXRE	LMAXRE
TR1	Horiz45	128ms	100ms	Upperlevel2	Lowerlevel2
TR1	Parabolic	Upperlevel2	Lowerlevel2	128hz	Lowerlevel3
TR2	Parabolic	Upperlevel2	Lowerlevel2	128hz	Lowerlevel3

SECTOR

SCANID	SECTORTYPE	UPERLIMITS	LOWERLIMITS	SECWAZ	SECWEL	SMTRACK
Sca1	Unidirectional	128ms	100ms	325ms	300kw	TR1
Sca2	Parabolic	Upperlevel2	Lowerlevel2	300kw	4ft	TR1
Sca2	Parabolic	Upperlevel2	Lowerlevel2	300kw	4ft	TR2

DIRECTION

RADPATID	ANTGAIN	SECCHAR	BWDTHAZ	BWDTHEL	FIRSTAZ	FIRSTEL
Rad1	10db	Sca1	325ms	300kw	4ft	325ms
Rad2	10db	Sca2	300kw	4ft	325ms	300kw
Rad3	10db	Sca2	300kw	4ft	325ms	300kw

Figure 5 (cont'd) Relational Database Instance of EWIR Schema

III. ACTIVATION OF THE RELATIONAL INTERFACE

It was not possible to ascertain with any degree of certainty the last time the relational interface to the M²DBMS was utilized. Development and implementation of the object-oriented interface has been the focus of research in recent years. Consequently, the other interfaces have lain dormant. This situation is exacerbated by the fact that there exists little documentation of changes made to either the system hardware or software.

A. STARTING UP THE INTERFACE

The first step in reactivating the relational interface was, predictably, locating the master source code for the interface.

The files are found in the following directory:

db11/u/mdbs/master/CNTRL/TI/LangIf/src/Sql/Lil

A complete file listing of the directory is provided in Figure 6.

Following the steps outlined in the User's Manual [Ref.3] the system was started in order to test the code. Interestingly, when the **start** command was entered the MDBS actually ran code under the following directory:

db11/u/mdbs/greg/CNTRL/TI/LangIf/src/Sql/Lil

Prudence dictated that a copy of the master source code be created to support our troubleshooting/debugging efforts. We placed the files in the

db11/u/mdbs/edwards/CNTRL/TI/LangIf/src/Sql/Lil
directory.

The User's Manual indicated that all the necessary data files for a simple schema load are in the **db11/u/mdbs/UserFiles** directory. That particular directory is, to put it bluntly, a mess. It appears that

files are added but rarely deleted. A listing of those files is provided in Figure 7. We decided to begin testing with the **COURSE** sample database found there. The significant steps taken in activating and testing the interface are detailed below. The User's Manual [Ref. 3.] provides more detail on this and all of the interfaces to the system.

After logging on to the system, but prior to executing the **run** command, it is necessary to verify that there are no processes still running the MDBS system. The UNIX command **ps ax** displays all the active processes and the command **kill** is used to eliminate the extraneous processes. Once this has been accomplished and the data disk zeroed, the MDBMS system is restarted using the **run** command. However, since the system is hard-coded to check the **UserFiles** directory, it is required that all the schema and request files be resident in that directory. A subdirectory of **UserFiles** can be used but the path name must be included when the system asks for a schema or request file name. The first system prompt asks the user to select the appropriate interface:

Select an operation:

- interface** (a) - Execute the attribute-based/ABDL
- (r) - Execute the relational/SQL interface
- interface** (h) - Execute the hierarchical/DL/I
- (n) - Execute the network/CODASYL interface
- interface** (f) - Execute the functional/DAPLEX
- (o) - Execute the Object-Oriented interface
- (x) - Exit to the operating system

At which juncture the user enters **r** to start the relational interface. At the next prompt we indicate that we will be loading a new database by entering **1**:

Enter type of operation desired

- (1) - load new database
- (p) - process existing database
- (x) - return to the MLDS/MBDS system menu

The next step is identifying the database you wish to work with:

Enter name of the database ---->

We, of course, enter **COURSE** at this point. It is significant to note that the database name must be in all capitals or the system will not recognize it. In order to proceed the user must then indicate the desired mode of input:

Enter mode of input desired

- (f) - read in a group of creates from a file
- (t) - read in creates from the terminal
- (x) - return to the main menu

It is a very good idea to use files rather than work from the terminal. To do so otherwise is time consuming and rapidly becomes frustrating. Obviously, then, we enter **f** to indicate that we have a file which is followed by the prompt:

What is the name of the CREATE/QUERY file ---->

We designated the **COURSEsqldb** file (Figure 8). The schema filename, by the way, must be in the format <database name><sqldb> for the relational interface. This unfortunately resulted in a core dump. The system reported that it was unable to find the file. Since the file was verified to be present in the directory, it was apparent that there was a deeper problem.

B. TROUBLESHOOTING

The program's inability to recognize the **COURSEsqldb** file was a nagging problem. We tried every possible combination of file and path name at the prompt, to no avail. The "debugging" code imbedded in the program indicated

We are in the COMMON/utilities.c add_path

at each core dump. Fortunately, discussion with students working on 4 revealed that the **add_path** function should accept two parameters. That function in our files took only one parameter. We edited the following **.c** files to have the **add_path** function also accept a character array/string called **location: lil.c, mass_load.c, buildddl.c, buildreldesc.c, and r_catalog.c**. After recompiling the code we were able to successfully run and query the **COURSE** database.

C. A SUCCESSFUL RUN

A review of the rest of the program run may prove enlightening and lay the necessary foundation for our discussion of the implementation of the EWIR database.

Once the system was able to find the **COURSEsqldb** schema file, it automatically creates the template (Figure 9) and descriptor (Figure 10) files. These are referred to as the **.t** and **.d** files, respectively. The template file provides the specification of the relational database in an equivalent kernel database and this template creates the attribute-value pair used by the kernel system. The descriptor file provides the kernel system with a list of all the relations in the database. Then the MDBS system will parse the relational schema file and transform it into ABDL, the kernel data model

language. The system offers an opportunity to index attributes in the relation, if desired. As noted in the User's Manual, and borne out by our experience, indexing is not usually used. Once the database schema is loaded the system will offer the following prompt:

Enter type of operation desired

- (l) - load new database
- (p) - process existing database
- (x) - return to the MLDS/MBDS system menu

Now we select option **p** to process our now existent database and proceed to load our data. If the system is re-entered after a database has been successfully loaded, **p** must be selected or an error message will appear.

Enter mode of input desired

- (f) - read in a group of queries from a file
- (t) - read in queries from the terminal
- (m) - mass load a file
- (d) - display the current database schema
- (x) - return to the previous menu

By far the most efficient way to load records, for all but the smallest databases, is the mass load function. It frees the user from having to write repetitive **INSERT** transactions in **SQL**, although this remains an option. All mass load files must be prefaced by the database name and have a **.r** suffix. The **COURSE.r** file is presented in Figure 11. This file name is entered at the prompt:

Enter name of record file ---->

The database is now ready to handle queries. Queries in both this sample database and the EWIR database are discussed in Chapter V.

A script of a successful test run of the **COURSE** database, complete with debugging comments, is provided in Enclosure 1. This enables us to move on to the next step in our project: implementation of the relational EWIR database. Chapter IV discusses the details of this implementation.

```
Makefile
buildddl.c
buildddl.c.BAK
buildddl.o
buildreldesc.c
buildreldesc.c.BAK
buildreldesc.o
dli_init.c
dli_init.c.BAK
flags.def
flags.def.BAK
lil.c
lil.c.BAK
lil.o
lil_main.c
lil_main.c.BAK
lil_main.o
lilcommon.c
lilcommon.c.BAK
lilcommon.o
```

```
mass_load.c
mass_load.c.BAK
mass_load.c.bak
mass_load.o
newuser.c
newuser.c.BAK
newuser.o
r_catalog.c
r_catalog.c.BAK
r_catalog.o
r_catdb.c
r_catdb.c.BAK
r_catdb.o
readrtnes.c
readrtnes.c.BAK
readrtnes.o
utilities.c
utilities.c.BAK
utilities.o
```

Figure 6 File Listing From
db11/u/mdbs/master/CNTRL/TI/LangIf/src/Sql/Lil

??s4??????	EWIRTsqldb*	MAINTdapreq1	SCHEDsqldb
A.r*	EWIR_ABDL#1	MAINTdapreq2	SCHOOL.s
AGE.s	EWIR_ABDL#2	MANT.d	T.t
AMMO#2	EWIR_ABDL#3	MANT.t	TEST#1
AMMO.d	EWIR_ABDL#4	NET1.d	TEST.d
AMMO.r	EWIR_ABDL.d	NET1.t	TEST.r*
AMMO.t	EWIR_ABDL.r	NET1dml db	TEST.t
AMMO1.d	EWIR_ABDL.t	NET1req	TEST2.d
AMMO1.r	EWIR_FDM.d	NEWIR.d	TEST2.r*
AMMO1.t	EWIR_FDM.t	NEWIR.r	TEST2.t
AMMOabdlreq#1	EWIR_FDMdapdb	NEWIR.t	TEST2dml db
ASP.d	EWIR_SQLsqldb	NOTES	TESTL#1
ASP.r	EWIRdml db	NPS.d	TESTdml db
ASP.t	EWIRsqldb*	NPS.r	VEHICLES.d
ASP1.r	EWIRsqlreq1*	NPS.t	VEHICLES.r
ASPabdlreq#1	EWIRsqlreq2*	NUMBER.S	VEHICLES.r~
Asqldb*	EWIRsqlreq3*	ORIGDEWIR.r	VEHICLES.t
COFOID_c	EWIRsqlreq4*	PART.d	VEHICLESooldb
COMPANY.d	EWIRsqlreq5*	PART.r*	VEHICLESoolins

Figure 7 File Listing from db11/u/mdbs/UserFiles

COMPANY.r	FAC.t	PART.t	VEHICLESoolreq
COMPANY.t	FACSTU#1	PARTS#1	VEHICLESsqlreq
COMPANY1.d	FACSTU#11	PARTS.d	VEHILCES.d
COMPANY1.t	FACSTU#12	PARTS.r*	VEHILCES.t
COURSE.d	FACSTU#13	PARTS.t	abdm/
COURSE.out*	FACSTU#14	PARTS1.d	bill.r
COURSE.r	FACSTU#15	PARTS1.t	brq
COURSE.t	FACSTU#16	PARTS2.d	ct
COURSEsqldb	FACSTU#17	PARTS2.r*	daplex/
COURSEsqlreq1	FACSTU#1~	PARTS2.t	data_files
COURSEsqlreq2	FACSTU#20	PARTS2dmldb	ddb
COURSEsqlreq4	FACSTU.d	PARTS3.d	ddbq
COURSEsqlreq5	FACSTU.dict	PARTS3.r*	ddd#1
DEWIR.d	FACSTU.r	PARTS3.t	demo
DEWIR.r	FACSTU.t	PARTS3dmldb	demoreq
DEWIR.t	FACSTU2.r	PARTS4.d	dmldb1
DEWIRdmldb	FACSTUOOLDB.t	PARTS4.r*	dmlreq1
ELe	FACSTUooldb	PARTS4.t	done
EMPREC.d	FACSTUooldb~	PARTS4dmldb	dreq

Figure 7 (cont'd) File Listing from db11/u/mdbs/UserFiles

EMPREC.r	FACSTUoolreq1	PARTS5.d	fambak/
EMPREC.t	FACSTUoolreq10	PARTS5.r	hierarchical/
EMPRECreq	FACSTUoolreq11	PARTS5.t	hoppalal
EMPRECsqldb	FACSTUoolreq12	PARTS5dmlldb	n#1
ETC/	FACSTUoolreq1~	PARTSdmlldb	netreq
EWIR#1	FACSTUoolreq2	RAMIREZ.d	network/
EWIR.d	FACSTUoolreq2~	RAMIREZ.dict	new*
EWIR.r*	FACSTUoolreq3	RAMIREZ.t	o-o/
EWIR.t	FACSTUoolreq4	RECEPT.d	oldFACSTUoolreq2
EWIR1Areq*	FACSTUoolreq5	RECEPT.dict	ool_queryfile
EWIR1E.d	FACSTUoolreq6	RECEPT.t	ool_queryfile~
EWIR1E.r*	FACSTUoolreq6~	RECEPT2.d	out.1*
EWIR1E.t	FACSTUoolreq7	RECEPT2.dict	out.2*
EWIR1EA.r*	FACSTUoolreq7~	RECEPT2.t	out.3*
EWIR1EAsqldb*	FACSTUoolreq8	RECEPT3.d	out.4*
EWIR1Ereq*	FACSTUoolreq9	RECEPT3.dict	p2req
EWIR1Ereq.bak*	FAMILY.d	RECEPT3.t	p3req
EWIR1Ereq1*	FAMILY.r	RECEPT4.d	p4req
EWIR1Ereq2*	FAMILY.t	RECEPT4.dict	p5req
EWIR1Esqldb*	FAMILY.t.backup	RECEPT4.t	preq
EWIR2E.r*	FAMILYooldb	SALES#1	query9
EWIR2Ereq*	FAMILYoolreq	SALES#100	query_1
EWIR2Esqldb*	FAMILYoolreq2	SALES#2	query_10
EWIRA.d	FEWIR.d	SALES#22	query_2
EWIRA.r*	FEWIR.r	SALES#2a	query_3

Figure 7 (cont'd) File Listing from dB11/u/mdbs/UserFiles

EWIRA.t	FEWIR.t	SALES#2b	query_4
EWIRAsqlldb*	FEWIRdmlldb	SALES#2c	query_5
EWIRB.d	FEWIRreq	SALES#3	query_6
EWIRB.r*	FLEET.r	SALES#33	query_7
EWIRB.t	FLEETooldb	SALES#4	query_8
EWIRBsqlldb*	FLEEToolreq	SALES#5	query_9
EWIRC.r*	FLEETsqlreq	SALES#6	query_f
EWIRCsqlldb*	FNAME.s	SALES#7	r
EWIROODB.d	FRIENDS.s	SALES#8	relational/
EWIROODB.dbak	FSON#1	SALES#9	req1
EWIROODB.dict	FSON#1~	SALES.d	req2
EWIROODB.dictbak	FSON.d	SALES.r	rq1
EWIROODB.r	FSON.old	SALES.t	rq2
EWIROODB.r1	FSON.r	SALES100	rq3
EWIROODB.rbak	FSON.r~	SALES2.r	rq33
EWIROODB.t	FSON.t	SALES2.r.BAK	rq4
EWIROODB.tbak	FSON.t~	SALES3.r	s_and_f_files/
EWIROOBDTEMP.r	Ff	SALESabdlreq1	temp.txt
EWIROOBDold.r	HOPPALA	SCHEDULE.d	tempjunk/
EWIROODL	JOB.s	SCHEDULE.r	test*
EWIROODL_org	LNAME.s	SCHEDULE.t	treq
EWIROODLbak	MAINT.d	SCHEDULEsqlldb	x
EWIROODLold	MAINT.r	SCHEDULEsqlreq	xedit
EWIROODL~	MAINT.t	SCHEDreq	zVEHICLE.r
EWIRT.d	MAINT2.r	SCHEDreq1	zVEHICLEoolddb
EWIRT.r*	MAINTdapdb	SCHEDreq2	
EWIRT.t	MAINTdapreq	SCHEDreq3	

Figure 7 (cont'd) File Listing from db11/u/mdbs/UserFiles

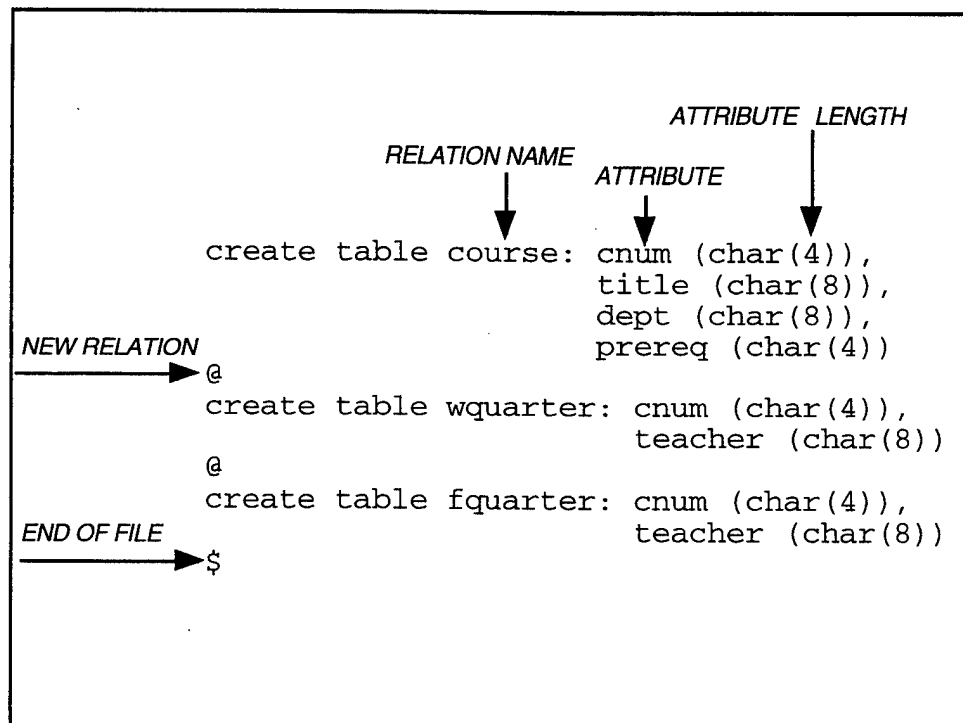


Figure 8 COURSE Database Schema Specification

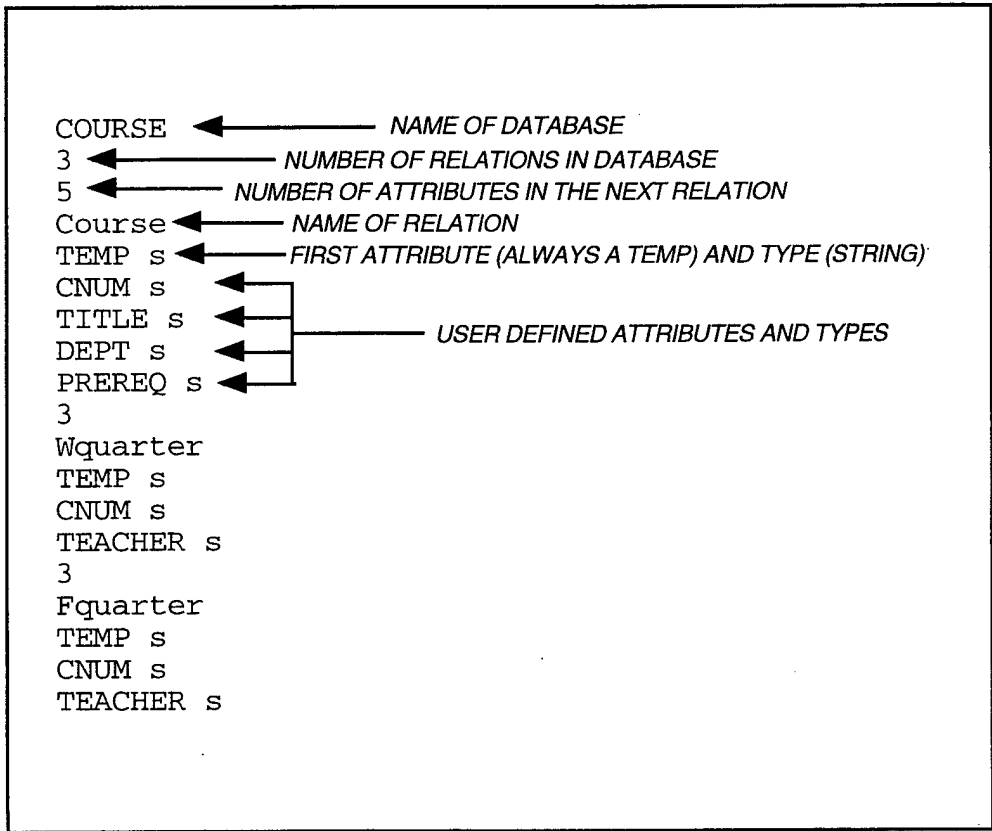


Figure 9 The COURSE Template File Format

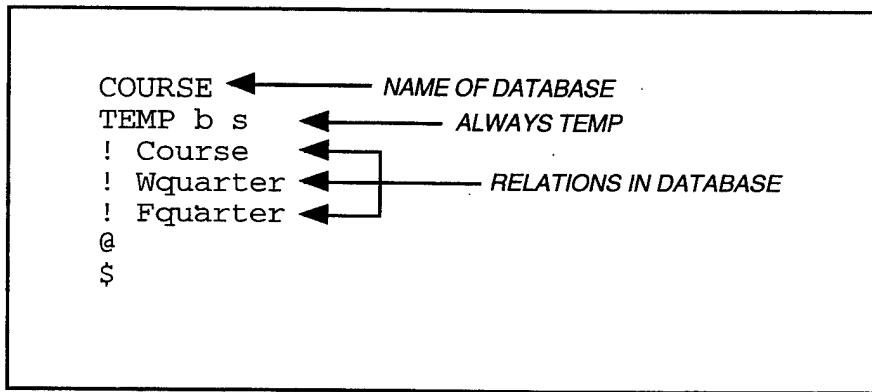


Figure 10 The COURSE Descriptor File

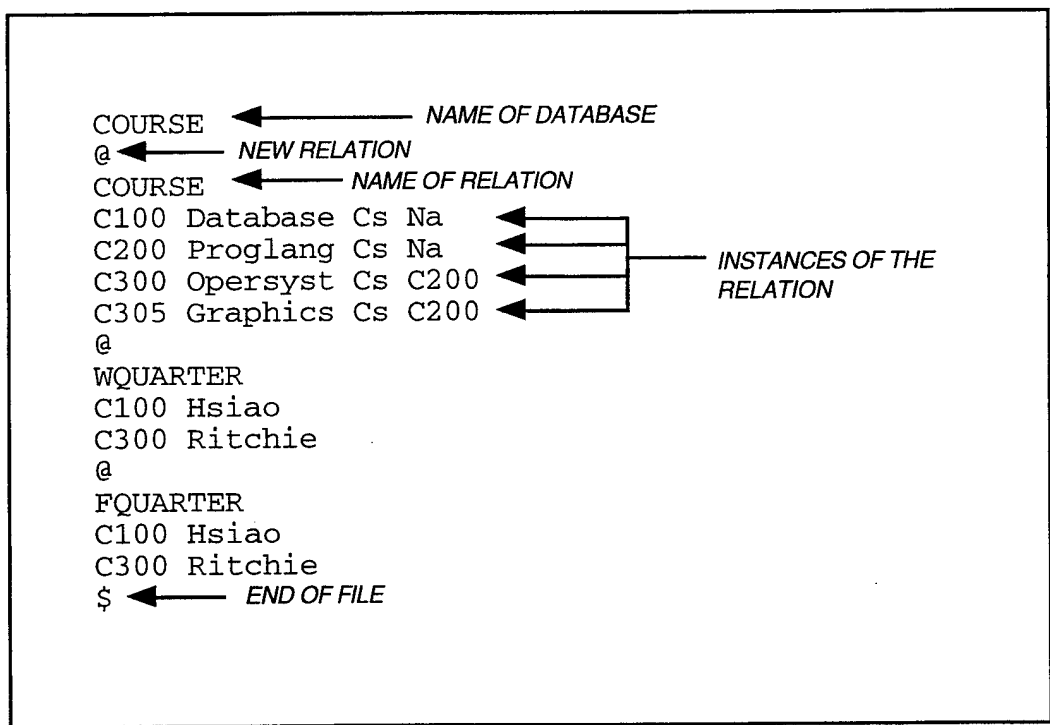


Figure 11 The COURSE Record File

IV. IMPLEMENTATION OF THE EWIR DATABASE

Prior to attempting implementation of the EWIR database, it was imperative that we first be able to consistently and reliably load and query the sample databases. We conducted tests with both the **COURSE** database discussed in the previous section, as well as the **EMPREC** database detailed in the User's Manual [Ref.3]. In the course of our troubleshooting, it was particularly helpful to be able to periodically verify that the system as a whole was still functioning normally. This enabled us to attribute any difficulties encountered to problems in the actual EWIR database and related files rather than to the source code.

A. THE EWIRsql**db** FILE

Chapter II outlines the transformation of the object-oriented EWIR database [Ref.2] into a relational model. This, of course, is only a preliminary step to implementation of the model in the M²DBMS. The schema (Figure 4) serves as a guide for drafting the **EWIRsql**db**** file which outlines the schema of the database in SQL. As discussed in Chapter III, this is the file which permits the Language Interface Layer to generate the **.d** and **.t** files. The User's Manual [Ref.3] provided few specifics on syntax, other than the statement that files which contain multiple transactions must have an "@" sign between each transaction and all files must have an end of file marker, "\$", on the last line of the file. In the absence of detailed guidance, we attempted to mimic the format of the **COURSEsql**db**** sample file (Figure 8) as closely as possible. This, in conjunction with the command of SQL developed in class, facilitated writing of our first draft which, naturally, bombed. A lengthy period of trial and error revealed the following apparent syntax restrictions:

- names of relations (tables) are limited to ten characters
- lower case only throughout the file
- underscores are not permitted in attribute names
- attribute names are limited to fifteen characters
- within each **create table** attribute names are separated by commas
- the end of file marker, "\$", must be followed by a carriage return or the system will crash

Our efforts at fixing this particular file were frustrated by our persistent inability to load a corrected copy of the schema file. We eventually discovered a file

db11/u/mdbs/edwards/run/.sql.db1

which is apparently updated when the schema is loaded. It contains only the line **EWIR COURSE EMPREC** and seems to indicate which databases have been loaded. In order for the system to accept a corrected **EWIRsqldb** file, **EWIR** must first be removed manually from the **.sql.db1** file. Figure 12 is the EWIR database schema specification which finally worked. Figures 13 and 14 are the template and descriptor files, respectively, which are generated by the system once **EWIRsqldb** is loaded. A detailed description of the elements of these files is provided in Figures 8, 9 and 10 and will not be repeated here.

B. MASS LOADING THE DATA

Efforts to mass load the database instances bombed repeatedly. As mentioned earlier, our confidence in the system enabled us to largely discount the source code as the culprit. Through systematic trial and error we discovered a number of syntax problems with our attempt to write the record, or **EWIR.r**, file. The following is a summary of the additional restrictions we discovered:

- the database and relation names must be all uppercase
- although attribute names may not contain underscores, the data may
 - the first character of each data instance must be capitalized
 - particular care must be paid to ensure that the spelling of relation and attribute names from file to file remains identical

In addition to the syntax difficulties, it took us some time to realize that, while the **.t** and **.d** files are generated automatically, they must be manually transferred to **db13** in order for the mass load function to work. Figure 15 is the **EWIR.r** file which successfully mass loads the data. A script file which documents loading the EWIR schema, complete with debugging statements is provided in Enclosure 2.

At this point we have successfully loaded both a relational schema and a database instance for EWIR. A database, however, is largely useless unless it can provide meaningful replies to user queries. Chapter V discusses the challenges we encountered in querying the EWIR database.

```

create table emitter: uniqueid (char(4)),
                    weaponsystem (char(6)),
                    emitfunction (char(15)),
                    emitptfgen (char(15)),
                    erfeccm (char(4))

@
create table antenna: antennaid (char(3)),
                    anttype (char(13)),
                    antfunct (char(13)),
                    hordim (char(6)),
                    vertdim (char(6)),
                    antdirec (char(5)),
                    acelpol (char(3)),
                    uniqueid (char(4))

@
create table direction: radpatid (char(5)),
                    antgain (char(5)),
                    secchar (char(5)),
                    bwdthaz (char(6)),
                    bwdthel (char(6)),
                    firstaz (char(6)),
                    firstel (char(6))

@
create table circorelip: polarid (char(3)),
                    polardata (char(7)),
                    sense (char(10)),
                    axratio (char(6)),
                    comment (char(14)),
                    contagency (char(9)),
                    lastupdate(char(5))

@

```

Figure 12 EWIR Database Schema Specification
(cont'd next page)

```

create table mechscan: scanid (char(5)),
                        smpavgtime (char(6)),
                        thresholdmeas (char(6)),
                        planscan (char(15)),
                        stpsgability (char(12)),
                        scfunct (char(12)),
                        radpatid (char(5))

@
create table mechtrack: trackid (char(4)),
                        plantrack (char(10)),
                        umaxra (char(11)),
                        lmaxra (char(11)),
                        umaxre (char(11)),
                        lmaxre (char(11))

@
create table sector: scanid (char(5)),
                    sectortype (char(9)),
                    upperlimits (char(11)),
                    lowerlimits (char(11)),
                    secwaz (char(6)),
                    secwel (char(6)),
                    smtrack (char(4))

$

```

Figure 12 EWIR Database Schema Specification

```

EWIR
7
6
Emitter
TEMP s
UNIQUEID s
WEAPONSYSTEM s
EMITFUNCTION s
EMITPTFGEN s
ERFECCM s
8
Antenna
TEMP s
ANTENNAID s
ANTTYPE s
ANTFUNCT s
HORDIM s
VERTDIM s
ANTDIREC s
ACELPOL s
8
Direction
TEMP s
RADPATID s
ANTGAIN s
SECCHAR s
BWDTHAZ s
BWDTHEL s
FIRSTAZ s
FIRSTEL s
8
Circorelip
TEMP s
POLARID s
POLARDATA s
SENSE s
AXRATIO s
COMMENT s
CONTAGENCY s
LASTUPDATE s

```

```

7
Mechscan
TEMP s
SCANID s
SMPAVGTIME s
THRESHOLDMEAS s
PLANSCAN s
STPSGABILITY s
SCFUNCT s
7
Mechtrack
TEMP s
TRACKID s
PLANTRACK s
UMAXRA s
LMAXRA s
UMAXRE s
LMAXRE s
8
Sector
TEMP s
SCANID s
SECTORTYPE s
UPPERLIMITS s
LOWERLIMITS s
SECWAZ s
SECWEL s
SMTRACK s

```

Figure 13 EWIR Database Template File

```
EWIR
TEMP b s
! Emitter
! Antenna
! Direction
! Circorelip
! Mechscan
! Mechtrack
! Sector
@
$
```

Figure 14 The EWIR
Descriptor File

```

EWIR
@
EMITTER
Ee1 Aa10 Parabolic Modpulsewave Ww1
Ee2 Aa6 Modpulsewave Phasedarray Ww2
Ee3 Sa21 Phasedarray Parabolic Ww3
Ee4 Aa9 Goofy Minnie Ww1
@
ANTENNA
Aa1 Phasedarray Longrngaa 3ft 4ft Rad1 Pp1 Ee1
Aa2 Squaresail Longrngaa 3ft 4ft Rad2 Pp2 Ee2
Aa3 Parabolic Longrngaa 325ms 300kw Rad2 Pp3 Ee3
@
DIRECTION
Rad1 10db Sca1 325ms 300kw 4ft 325ms
Rad2 10db Sca2 300kw 4ft 325ms 300kw
Rad3 10db Sca2 300kw 4ft 325ms 300kw
@
CIRCORELIP
Pp1 Cross Left 20db Cm1 Airforce 8_95
Pp2 Pdata2 Parabolic 300kw Satellite_det Airforce 8_95
Pp3 Pdata3 Parabolic 20db Satellite_det Airforce 8_95
@
MECHSCAN
Sca1 325ms 4ft Phasedarray Phasedarray Parabolic Rad1
Sca2 300kw 325ms Parabolic Parabolic Modpulsewave Rad2
Sca3 300ms 325kw Parabolic Parabolic Modpulsewave Rad2
@
MECHTRACK
TR1 Horiz45 128ms 100ms Upperlevel2 Lowerlevel2
TR1 Parabolic Upperlevel2 Lowerlevel2 128hz Lowerlevel3
TR2 Parabolic Upperlevel2 Lowerlevel2 128hz Lowerlevel3
@
SECTOR
Sca1 Unidirectional 128ms 100ms 325ms 300kw TR1
Sca2 Parabolic Upperlevel2 Lowerlevel2 300kw 4ft TR1
Sca2 Parabolic Upperlevel2 Lowerlevel2 300kw 4ft TR2
$

```

Figure 15 The EWIR Record File

V. SQL TRANSACTIONS

Thus far, in addition to presenting a general description of M²DBMS and how it works, we have both introduced the EWIR database and described the development of our relational model from a subset of that database. The actual implementation of that relational model was described in the previous chapter. In this chapter we discuss manipulation of the database, specifically through queries. As previously mentioned, our system utilizes a subset of SQL, or Structured Query Language, for both data definition and data manipulation. Basic understanding of the SQL language is presumed.

It was our expectation that, once the EWIR database was successfully implemented, we would be able to simply query the database following the guidance in the User's Manual [Ref.3]. Instead we discovered that the relational interface has some significant limitations, which we will discuss in detail. In order to place those findings in perspective, a more ideal example, querying the sample COURSE database introduced in Chapter III, is first discussed.

A. QUERYING THE COURSE DATABASE

Once the **COURSE.r** record file has been successfully mass-loaded the following prompt appears:

Enter mode of input desired

- (f) - read in a group of queries from a file
- (t) - read in queries from the terminal
- (m) - mass load a file
- (d) - display the current database schema
- (x) - return to the previous menu

While the system can handle ad hoc queries entered at the

terminal, in almost all cases it is preferable to have all desired queries prepared in a request file. This file is written in the appropriate data model language, in our case SQL, and is similar in format to other files with multiple transactions: each transaction must be separated by a "@" sign and the file must have an end of file marker, "\$", on the last line followed by a carriage return. Interestingly, while the User's Manual states that the request file name must include the suffix **req** our experience does not bear this out, although for the sake of consistency it is a good idea. The **COURSEreq** file is shown in Figure 16. To proceed with this file the user selects **f** at this juncture. The queries will appear on the terminal and be followed by the prompt:

Pick the number or letter of the action desired

- (num) - execute one of the preceding queries
- (d) - redisplay the file of queries
- (x) - return to the previous menu

Any of the displayed queries may be selected at this point by entering the corresponding number and the answer will appear on the screen. For example, when number 1 is entered the following response appears:

CNUM	TITLE	DEPT	PREREQ
C100	Database	Cs	Na
C200	Proglang	Cs	Na
C300	Opersyst	Cs	C200
C305	Graphics	Cs	C200

followed immediately by the now familiar prompt:

Pick the number or letter of the action desired

- (num) - execute one of the preceding queries

- (d) - redisplay the file of queries
- (x) - return to the previous menu

The user can select any or all of the queries in the **req** file in this fashion. A complete script which includes all of the queries and their answers can be found in Enclosure 1. The extensive debugging statements are helpful in understanding the mechanics of the program.

B. QUERIES OF THE EWIR DATABASE

As was explained at the outset of this thesis, it is our goal to mirror the work of Reference 4, the reactivation of the network interface, as well as the implementation of the EWIR database in the object-oriented interface [Ref.2]. In order to balance the two goals, a mutually agreed upon list of queries was developed. It was all of our intents to exercise fully the capability of both the network and relational interfaces. The queries, simply stated in English, are:

1. Find an antenna with long-range anti-air capability.
2. Find the antenna type and function of an antenna with cross polarization.
3. Find a long-range antenna with a tracking plane of 45 degrees horizontal.
4. Find a long-range antenna that is not cross-polarized and has a sector scan that is parabolic.
5. Find the corresponding weapon system and emitter ID for query number 4.
6. Find the sector type and upper and lower values of the period limits of an antenna with an average sample scan time of 325 ms.

7. Find the weapon system of an antenna with an antenna type of square-sail and an ax-ratio of 300 kw.

Initially we simply translated these queries into SQL and wrote an **EWIRreq** file which promptly bombed. This necessitated a laborious search to discover the source of the problem, the details of which are presented in the following section.

C. QUERY LIMITATIONS OF THE RELATIONAL INTERFACE

After ruling out syntax as the source of the difficulties in querying the EWIR database, we focused on the size of the database. Recall that the **EWIRsqldb** file discussed in Chapter IV (Figure 12) consisted of seven relations with between five and eight attributes each. We could find no documentation of the processing limitations of the relational interface and no arbitrary limits set in the source code. Our only recourse was to systematically vary the number of relations and the number of attributes per relation. A list of all the permutations we attempted would not be particularly enlightening so, suffice it to say, we eventually found what we believe to be the upper limits of the interface. In the final analysis, we did not exceed any maximum number of attributes with the EWIR relations but the system does not appear to be able to handle any more than four relations in a database. Not surprisingly, both the EMPREC and COURSE sample databases had only three relations each.

In order to proceed with processing our queries, then, it was necessary to develop a set of sub-databases with four or fewer relations in each. Of course, before we could decide which relations to exclude we had to first determine

which relations are required for each query. Ultimately four sub-databases, EWIR1E, EWIR2E, EWIR3E and EWIR4E (Figures 17 through 20, respectively), are specified. Figures 21 through 24 are the record, or **.r**, files which correspond to each of the sub-databases.

One other frustrating discovery was the impact of the limited available memory on querying. It is not unusual for the system to experience a memory dump after running several complex queries. We were unable to quantify this observation but we became very conservative about the number of complex queries we would attempt prior to backing out and starting over. This phenomenon was not noted when working with either of the sample databases.

D. THE EWIR DATABASE QUERIES

In the interest of clarity, the queries will be discussed in the order they are listed in Section B and the following format will be used:

Subsection # <Query number>

Query in English:

Query in SQL:

Applicable sub-database:

Query result:

Comments:

1. Query 1

Query in English: Find an antenna with long-range anti-air capability.

Query in SQL:

```

select antennaid, anttype, antfunct
from antenna
where antfunct = 'Longrngaa'

```

Applicable sub-database: EWIR1E (Figure 17)

Query result:

```

Q1
ANTENNAID | ANTTYPE           | ANTFUNCT   |
-----|-----|-----|
Aa1       | Phasedarray       | Longrngaa  |
Aa2       | Squaresail        | Longrngaa  |
Aa3       | Parabolic         | Longrngaa  |

```

Comments: When this type of select-project query is processed in the object-oriented database the information from a single antenna is returned. The relational interface, however, returns all of the antennas for which the specified conditions are met. The inclusion of antenna function in the result is an artificiality introduced to verify that the results obtained are valid. That information is not requested in the query.

2. Query 2

Query in English: Find the antenna type and function of an antenna with cross polarization.

Query in SQL:

```

select antennaid, anttype, antfunct
from antenna
where acelpol in
      (select polarid
       from circorelip

```

where polardata = 'Cross')

Applicable sub-database: EWIR1E (Figure 17)

Query result:

```
Q2
ANTENNAID |ANTTYPE          |ANTFUNCT          |
-----|-----|-----|
Aa1       |Phasedarray      |Longrngaa         |
```

Comments: It would have been more concise to write this query as follows:

```
select antennaid, anttype, antfunct
from antenna D, circorelip C
where D.acelpol = C.polarid and C.polardata = 'Cross'
```

This form of the query, however, bombs. It appears that the system is incapable of processing select-project-join type queries. As demonstrated above, this limitation can be circumvented by use of nesting.

3. Query 3

Query in English: Find a long-range antenna with a tracking plane of 45 degrees horizontal.

Query in SQL:

```
select antennaid, anttype, antfunct
from antenna
where antfunct = 'Longrngaa' and antdirec in
  (select radpatid
   from mechscan
   where scanid in
```

```

(select scanid
 from sector
 where smtrack in
 (select trackid
 from mechtrack
 where plantrack = 'Horiz45'))

```

Applicable sub-database: EWIR3E (Figure 19)

Query result:

Q3

ANTENNAID	ANTTYPE	ANTFUNCT
Aa1	Phasedarray	Longrngaa
Aa2	Squaresail	Longrngaa
Aa3	Parabolic	Longrngaa

Comments: As with the previous query, there are undoubtedly more concise ways to formulate this query. Unfortunately they all require joining more than one table in the **from** clause. To reiterate, the system does not currently support that type of join and therefore must be formulated using several layers of nesting.

4. Query 4

Query in English: Find a long-range antenna that is not cross-polarized and has a sector scan that is parabolic.

Query in SQL:

4.1.

```

select acelpol, antdirec, antennaid

```



```

from antenna
where antdirec in
  (select radpatid
   from direction
   where secchar in
     (select scanid
      from sector
      where sectortype = 'Parabolic'))

```

@

4.2

```

select antennaid, acelpol, antdirec
from antenna
where acelpol in
  (select polarid
   from circorelip
   where polardata /= 'Cross')

```

Applicable sub-database: EWIR1E (Figure 17)

Query result:

Q4.1

ACELPOL	ANTDIREC	ANTENNAID
Pp2	Rad2	Aa2
Pp3	Rad2	Aa3

Q4.2

ANTENNAID	ACELPOL	ANTDIREC
Aa2	Pp2	Rad2
Aa3	Pp3	Rad2

Comments: This query could not be fully processed. If the

query is divided into two component parts, as above, each half will yield a result, as demonstrated. In other words, it can find an antenna that has a sector scan that is parabolic and it can find an antenna that is not cross-polarized but it can't find one that meets both criteria. A query that meets both would look like:

```
select antennaid, acelpol, antdirec
from antenna
where acelpol in
    (select polarid
     from circorelip
     where polardata != 'Cross')
and
antdirec in
    (select radpatid
     from direction
     where secchar in
        (select scanid
         from sector
         where sectortype = 'Parabolic'))
```

The system difficulty with this query appears to be the amount of memory it demands. When this query is selected the system experiences a memory dump.

5. Query 5

Query in English: Find the corresponding weapon system and emitter ID for query number 4.

Query in SQL: see comments

Applicable sub-database: none applicable, see comments

Query result: see comments

Comments: This query requires five tables to process and therefore exceeds the capabilities of the relational interface at this time.

6. Query 6

Query in English: Find the sector type and upper and lower values of the period limits of an antenna with an average sample scan time of 325 ms.

Query in SQL:

```
select sectortype, upperlimits, lowerlimits
from sector
where scanid in
  (select scanid
   from mechscan
   where smpavgtime = '325ms')
```

Applicable sub-database: EWIR2E (Figure 18)

Query result:

```
Q6
SECTORTYPE | UPPERLIMITS | LOWERLIMITS |
-----|-----|-----|
Unidirectional|128ms      |100ms      |
```

Comments: In SQL this query is very similar to query 2 and is included largely for the capabilities it demonstrates in the network interface [Ref.4].

7. Query 7

Query in English: Find the weapon system of an antenna with an antenna type of square-sail and an ax-ratio of 300 kw.

Query in SQL:

```
select uniqueid, weaponsystem
from emitter
where uniqueid in
  (select uniqueid, antennaid
   from antenna
   where anttype = 'Squaresail'
    and acelpol in
      (select polarid
       from circorelip
       where axratio = '300kw'))
```

Applicable sub-database: EWIR4E (Figure 20)

Query result:

```
Q7
UNIQUEID | WEAPONSYSTEM |
-----|-----|
Ee2      | Aa6           |
```

Comments: Once again, this query could be simplified greatly by using joins, which the system does not support, rather than the extensive nesting above.

E. SQL QUERY LIMITATIONS IN M²DBMS

SQL is an excellent choice for a data manipulation language since it is generally easy to understand and master. The fact that it also serves as a data definition language enhances its appeal. The problem with SQL in M²DBMS is that the full range of functionality is not available. As mentioned previously, the system does not allow joins. In other words, the **from** statement may only specify a single table. While we are able to use nesting to extract data, queries can rapidly become cumbersome when more than two levels of nesting are required.

Another significant limitation of the system is the allocated memory. A database that can not contain more than four relations is obviously not tenable and the inability to process complex queries, such as query 4, is particularly bothersome. The relational interface is, however, sufficiently capable to serve as a foundation for the development of the cross-model access capability.

```

select *
from course
@
select *
from fquarter
@
select *
from wquarter
@
select *
      from course
      where prereq /= 'na'
@
select cnum,prereq
      from course
      where prereq /= 'na'
      or cnum = 'C100'
@
select cnum,title,dept,prereq
from course
where cnum in
      (select cnum
       from wquarter)
@
select cnum,title,dept,prereq
from course
where cnum not in
      (select cnum
       from wquarter)
@
select
course.title,course.dept,wquarter.teacher
from course,wquarter
where course.cnum = wquarter.cnum
@
select cnum, prereq
from course
where cnum not in
      (select cnum
       from course
       where prereq = 'na')
$

```

Figure 16 SQL Request File COURSEreq

```

create table antenna: antennaid (char(3)),
                    anttype (char(13)),
                    antfunct (char(10)),
                    hordim (char(6)),
                    vertdim (char(6)),
                    antdirec (char(5)),
                    acelpol (char(3)),
                    uniqueid (char(4))

@
create table direction: radpatid (char(5)),
                    antgain (char(5)),
                    secchar (char(5)),
                    bwdthaz (char(6)),
                    bwdthel (char(6)),
                    firstaz (char(6)),
                    firstel (char(6))

@
create table circorelip: polarid (char(3)),
                    polardata (char(7)),
                    sense (char(10)),
                    axratio (char(6)),
                    comment (char(14)),
                    contagency (char(9)),
                    lastupdate(char(5))

@
create table sector: scanid (char(5)),
                    sectortype (char(9)),
                    upperlimits (char(11)),
                    lowerlimits (char(11)),
                    secwaz (char(6)),
                    secwel (char(6)),
                    smtrack (char(4))

$

```

Figure 17 EWIR Sub-Database Schema Specification 1

```

create table antenna: antennaid (char(3)),
                    anttype (char(13)),
                    antfunct (char(13)),
                    hordim (char(6)),
                    vertdim (char(6)),
                    antdirec (char(5)),
                    acelpol (char(3)),
                    uniqueid (char(4))
@
create table mechscan: scanid (char(5)),
                    smpavgtime (char(6)),
                    thresholdmeas (char(6)),
                    planscan (char(15)),
                    stpsgability (char(12)),
                    scfunct (char(12)),
                    radpatid (char(5))
@
create table sector: scanid (char(5)),
                    sectortype (char(9)),
                    upperlimits (char(11)),
                    lowerlimits (char(11)),
                    secwaz (char(6)),
                    secwel (char(6)),
                    smtrack (char(4))
$

```

Figure 18 EWIR Sub-Database Schema Specification 2


```

create table antenna: antennaid (char(3)),
                    anttype (char(13)),
                    antfunct (char(13)),
                    hordim (char(6)),
                    vertdim (char(6)),
                    antdirec (char(5)),
                    acelpol (char(3)),
                    uniqueid (char(4))

@
create table mechscan: scanid (char(5)),
                    smpavgtime (char(6)),
                    thresholdmeas (char(6)),
                    planscan (char(15)),
                    stpsgability (char(12)),
                    scfunct (char(12)),
                    radpatid (char(5))

@
create table mechtrack: trackid (char(4)),
                    plantrack (char(10)),
                    umaxra (char(11)),
                    lmaxra (char(11)),
                    umaxre (char(11)),
                    lmaxre (char(11))

@
create table sector: scanid (char(5)),
                    sectortype (char(9)),
                    upperlimits (char(11)),
                    lowerlimits (char(11)),
                    secwaz (char(6)),
                    secwel (char(6)),
                    smtrack (char(4))

$

```

Figure 19 EWIR Sub-Database Schema Specification 3

```

create table emitter: uniqueid (char(4)),
                    weaponsystem (char(6)),
                    emitfunction (char(15)),
                    emitptfgen (char(15)),
                    erfeccm (char(4))

@
create table antenna: antennaid (char(3)),
                    anttype (char(13)),
                    antfunct (char(13)),
                    hordim (char(6)),
                    vertdim (char(6)),
                    antdirec (char(5)),
                    acelpol (char(3)),
                    uniqueid (char(4))

@
create table circorelip: polarid (char(3)),
                    polardata (char(7)),
                    sense (char(10)),
                    axratio (char(6)),
                    comment (char(14)),
                    contagency (char(9)),
                    lastupdate(char(5))

$

```

Figure 20 EWIR Sub-Database Schema Specification 4

```

EWIR1E
@
ANTENNA
Aa1 Phasedarray Longrngaa 3ft 4ft Rad1 Pp1 Ee1
Aa2 Squaresail Longrngaa 3ft 4ft Rad2 Pp2 Ee2
Aa3 Parabolic Longrngaa 325ms 300kw Rad2 Pp3 Ee3
@
DIRECTION
Rad1 10db Sca1 325ms 300kw 4ft 325ms
Rad2 10db Sca2 300kw 4ft 325ms 300kw
Rad3 10db Sca2 300kw 4ft 325ms 300kw
@
CIRCORELIP
Pp1 Cross Left 20db Cm1 Airforce 8_95
Pp2 Pdata2 Parabolic 300kw Satellite_det Airforce 8_95
Pp3 Pdata3 Parabolic 20db Satellite_det Airforce 8_95
@
SECTOR
Sca1 Unidirec 128ms 100ms 325ms 300kw TR1
Sca2 Parabolic Upperlevel2 Lowerlevel2 300kw 4ft TR1
Sca2 Parabolic Upperlevel2 Lowerlevel2 300kw 4ft TR2
$

```

Figure 21 The EWIR1E Record File

EWIR2E

@

ANTENNA

Aa1 Phasedarray Longrngaa 3ft 4ft Rad1 Pp1 Ee1

Aa2 Squaresail Longrngaa 3ft 4ft Rad2 Pp2 Ee2

Aa3 Parabolic Longrngaa 325ms 300kw Rad2 Pp3 Ee3

@

MECHSCAN

Sca1 325ms 4ft Phasedarray Phasedarray Parabolic Rad1

Sca2 300kw 325ms Parabolic Parabolic Modpulsewave Rad2

Sca3 300ms 325kw Parabolic Parabolic Modpulsewave Rad2

@

SECTOR

Sca1 Unidirectional 128ms 100ms 325ms 300kw TR1

Sca2 Parabolic Upperlevel2 Lowerlevel2 300kw 4ft TR1

Sca2 Parabolic Upperlevel2 Lowerlevel2 300kw 4ft TR2

\$

Figure 22 The EWIR2E Record File

```

EWIR3E
@
ANTENNA
Aa1 Phasedarray Longrngaa 3ft 4ft Rad1 Pp1 Ee1
Aa2 Squaresail Longrngaa 3ft 4ft Rad2 Pp2 Ee2
Aa3 Parabolic Longrngaa 325ms 300kw Rad2 Pp3 Ee3
@
MECHSCAN
Sca1 325ms 4ft Phasedarray Phasedarray Parabolic Rad1
Sca2 300kw 325ms Parabolic Parabolic Modpulsewave Rad2
Sca3 300ms 325kw Parabolic Parabolic Modpulsewave Rad2
@
MECHTRACK
TR1 Horiz45 128ms 100ms Upperlevel2 Lowerlevel2
TR1 Parabolic Upperlevel2 Lowerlevel2 128hz Lowerlevel3
TR2 Parabolic Upperlevel2 Lowerlevel2 128hz Lowerlevel3
@
SECTOR
Sca1 Unidirectional 128ms 100ms 325ms 300kw TR1
Sca2 Parabolic Upperlevel2 Lowerlevel2 300kw 4ft TR1
Sca2 Parabolic Upperlevel2 Lowerlevel2 300kw 4ft TR2
$

```

Figure 23 The EWIR3E Record File

EWIR4E

@

EMITTER

Ee1 Aa10 Parabolic Modpulsewave Ww1

Ee2 Aa6 Modpulsewave Phasedarray Ww2

Ee3 Sa21 Phasedarray Parabolic Ww3

Ee4 Aa9 Goofy Minnie Ww1

@

ANTENNA

Aa1 Phasedarray Longrngaa 3ft 4ft Rad1 Pp1 Ee1

Aa2 Squaresail Longrngaa 3ft 4ft Rad2 Pp2 Ee2

Aa3 Parabolic Longrngaa 325ms 300kw Rad2 Pp3 Ee3

@

CIRCORELIP

Pp1 Cross Left 20db Cm1 Airforce 8_95

Pp2 Pdata2 Parabolic 300kw Satellite_det Airforce 8_95

Pp3 Pdata3 Parabolic 20db Satellite_det Airforce 8_95

\$

Figure 24 The EWIR4E Record File

VI. CONCLUSION

The purpose of this research was to lay a foundation for the development of cross-model access in the M2DEMS by reactivating the relational interface and implementing a subset of the EWIR database. In addition, it was our intention to discover and document problems in and limitations of the relational interface and make an assessment of the real-world suitability of the interface for EWIR.

A. SUMMARY

The relational interface is operational and can now be used as a basis for research in the cross-model access capability. In the process of reactivation we were able to document a number of syntactic restrictions as well as correct problems in the source code. A subset of the EWIR database implemented in the newly developed object-oriented interface was modeled and implemented in the relational interface. In the course of querying this new database several difficulties, mostly attributable to the size of the database and complexity of the queries, were documented. Of seven queries common to our and our sister thesis [Ref.4], five could be answered completely, one partially answered, and one query could not even be asked due to memory limitations. This list of queries and their results should prove edifying to follow-on researchers.

B. SUITABILITY OF THE RELATIONAL INTERFACE FOR EWIR

At the outset we did not anticipate that the relational interface was going to prove directly useful in implementing the EWIR database. We still believe that to be true. The

EWIR database, as previously discussed here and in References 1 and 2, is rich and complex. References 1 and 2 provide a thorough discussion of the advantages of the object-oriented model in making EWIR a more intuitive, natural, and powerful database. We see no point in attempting a full scale implementation of this database in any language that does not support inheritance.

C. FUTURE RESEARCH SUGGESTIONS

As mentioned several times throughout this thesis, the plan to implement the cross-model access capability of M2DBMS is underway. Our experience has suggested a few other courses of inquiry as well. One difficulty we encountered was the absence of a clear "timeline" of work on the system. As previously stated, this project has been ongoing for over a decade and no clear history of system changes exists. A document which provides those elements would be valuable to future researchers. Such centralized information may also foster better documentation of changes to the system's hardware and software.

Another recurring problem was a poor understanding of the limitations of the system's available memory, both for loading new databases and processing queries. Probing and documenting those limits is a possible topic for future work. We did not have the time to fully explore and document the impact of a larger database on other SQL data manipulation capabilities in the relational interface. While we reported the limitations we encountered in the course of our work, we believe that there may be more.

LIST OF REFERENCES

1. Coyne, K., *The Design and Analysis of an Object-Oriented Database of Electronic Warfare Data*, Master's Thesis, Naval Postgraduate School, Monterey, California, March 1996.
2. Lee, J.J. and McKenna, T.D., *The Object-Oriented Database and Processing of Electronic Warfare Data*, Master's Thesis, Naval Postgraduate School, Monterey, California, March 1996.
3. Bourgeois, P.A., *The Instrumentation of the Multimodel and Multilingual User Interface*, Master's Thesis, Naval Postgraduate School, Monterey, California, March 1993.
4. Werre, T.J. and Diehl, B.A., *The Activation and Testing of the Network CODASYL-DML Interface of the M²DBMS Using the EWIR Database*, Master's Thesis, Naval Postgraduate School, Monterey, California, June 1996.
5. Elmasri, R. and Navathe, S.B., *Fundamentals of Database Systems*, The Benjamin/Cummings Publishing Company, Inc., 1994.

APPENDIX A. COURSE SAMPLE DATABASE RUN SCRIPT

Welcome to MDBS, today is Mon May 6 13:56:09 PDT 1996

Check the time each file was last compiled:

```
-rwxrwxr-x 1 mdbs      262144 May 31 1995
../BE/NEWESTrecp.exe
-rwxrwxr-x 1 mdbs      98304 May 31 1995 ../BE/bget.exe
-rwxrwxr-x 1 mdbs      98304 May 31 1995 ../BE/bput.exe
-rwxrwxr-x 1 mdbs     204800 May 31 1995 ../BE/cc.exe
-rwxrwxr-x 1 mdbs      40960 May 31 1995 ../BE/dio.exe
-rwxrwxr-x 1 mdbs     262144 Jan 22 13:39 ../BE/dirman.exe
-rwxrwxr-x 1 mdbs     245760 Jan 22 13:39 ../BE/recp.exe

-rwxrwxr-x 1 mdbs     106496 May 31 1995
../CNTRL/cget.exe
-rwxrwxr-x 1 mdbs     106496 May 31 1995
../CNTRL/cput.exe
-rwxrwxr-x 1 mdbs     114688 Jan 22 13:38 ../CNTRL/iig.exe
-rwxrwxr-x 1 mdbs     966656 Jun  2 1995 ../CNTRL/o.exe
-rwxrwxr-x 1 mdbs     966656 Jan 22 13:50
../CNTRL/old4ti.exe
-rwxrwxr-x 1 mdbs     966656 Jan 12 11:47
../CNTRL/oldti2.exe
-rwxrwxr-x 1 mdbs     966656 Jan 16 11:15
../CNTRL/oldti3.exe
-rwxrwxr-x 1 mdbs     966656 Jan 31 00:34
../CNTRL/oldti4.exe
-rwxrwxr-x 1 mdbs     966656 Feb 29 16:01
```

```
../CNTRL/oldti5.exe
-rwxrwxr-x  1 mdbs          966656 Sep  4 1995
../CNTRL/oldtioriginal.exe
-rwxrwxr-x  1 mdbs          114688 Jan 22 13:38 ../CNTRL/pp.exe
-rwxrwxr-x  1 mdbs          180224 Jan 22 13:38
../CNTRL/recp.exe
-rwxrwxr-x  1 mdbs          966656 Apr 29 14:50 ../CNTRL/ti.exe
```

There should be 12 files listed, if not you need to recompile.

Do you need to recompile any executable and/or copy the 6 executable files to each Back End (bget, bput, cc, dio, dirman, recp.exe)? (y/n) n

The Current Configuration is:

Version Name: edwards

Controller: db11

1 Back End:

db13

WARNING: All data will be lost if you reconfigure

Do you wish to reconfigure the Back Ends? (y/n) n

Do you wish to use current database? (y/n) n

Zeroing backend meta disk on back end, db13...

File to zero = /dev/sd2c

Bytes to zero = 1000000

Bytes written...

102400

204800

307200

409600

512000

614400

716800

819200

921600

1000000

Zeroing backend data disk on back end, db13...

File to zero = /dev/sd4c

Bytes to zero = 716800

Bytes written...

102400

204800

307200

409600

512000

614400

716800

Removing CINBT and IIG AT tables on controller, db11...

Do you wish to run the Multi Modal, Multi Lingual,
Multi Backended Database System? (y/n) y

stopping processes on back end db13

no processes to kill on db13

stopping processes on db11, the controller

stop.db11: syntax error at line 13: `(' unexpected

EXECUTING: start.cntrl

starting 5 of 6 controller processes on db11...

EXECUTING: rsh db13 -n /u/mdbs/be.edwards/run.be &

EXECUTING: /u/mdbs/edwards/CNTRL/ti.exe 1

PID written to /u/mdbs/.ti.exe.pid

**** Unlink error: No such file or directory

No match.

Running backend on db13...

[1] 4843

[2] 4844

[3] 4845

[0] 0000 System configured for 1 backend(s).

[4] 4846

[5] 4847

[6] 4848

MBDS: Initializing communications...

Seconds remaining: 5 4 3 2 1

The Multi-Lingual/Multi-Backend Database System

Select an operation:

- (a) - Execute the attribute-based/ABDL interface
- (r) - Execute the relational/SQL interface
- (h) - Execute the hierarchical/DL/I interface
- (n) - Execute the network/CODASYL interface
- (f) - Execute the functional/DAPLEX interface
- (o) - Execute the Object-Oriented interface
- (x) - Exit to the operating system

Select-> r
Enter sql main
Enter new_rel_user
Exit new_rel_user
Enter r_language_interface_layer
Enter sql_init
Exit sql_init
Enter r_load_db_list
Exit r_load_db_list

Enter type of operation desired
 (l) - load new database
 (p) - process existing database
 (x) - return to the MLDS/MBDS system menu

[7;7mAction --- >[0;0m p

Enter r_process_old
[7;7m
Enter name of database ---->[0;0m COURSE
Enter r_load_catalog
Exit r_load_catalog

Enter mode of input desired
 (f) - read in a group of queries from a file
 (t) - read in queries from the terminal
 (m) - mass load a file
 (d) - display the current database schema
 (x) - return to the previous menu

[7;7mAction --- >[0;0m m

Selected Case m

Pass DBL_S\$Use

Pass TI_S\$AssignDBEnter mass_load

[7;7m

Enter name of record file ---->[0;0m COURSE.r

We are in the COMMON/utilities.c add_path

We have /u/mdbms/UserFiles/COURSE.r

Record file opened.

.Transfile opened.

Before going to to_caps, db_name is COURSE

db_name is COURSE

rdn_name is COURSE

[INSERT (<TEMP, Course>, <CNUM, C100>, <TITLE, Database>,
<DEPT, Cs>, <PREREQ, Na>)]

[INSERT (<TEMP, Course>, <CNUM, C200>, <TITLE, Proglang>,
<DEPT, Cs>, <PREREQ, Na>)]

[INSERT (<TEMP, Course>, <CNUM, C300>, <TITLE, Opersyst>,
<DEPT, Cs>, <PREREQ, C200>)]

[INSERT (<TEMP, Course>, <CNUM, C305>, <TITLE, Graphics>,
<DEPT, Cs>, <PREREQ, C200>)]

[INSERT (<TEMP, Wquarter>, <CNUM, C100>, <TEACHER, Hsiao>)]

[INSERT (<TEMP, Wquarter>, <CNUM, C300>, <TEACHER, Ritchie>)]

[INSERT (<TEMP, Fquarter>, <CNUM, C100>, <TEACHER, Hsiao>)]

[INSERT (<TEMP, Fquarter>, <CNUM, C300>, <TEACHER, Ritchie>)]

Exit mass_load

Enter mode of input desired

- (f) - read in a group of queries from a file
- (t) - read in queries from the terminal
- (m) - mass load a file
- (d) - display the current database schema
- (x) - return to the previous menu

[7;7mAction --- >[0;0m f

Enter r_read_transaction_file

[7;7m

What is the name of the CREATE/QUERY file ---->[0;0m

COURSEsqlreq1

We are in the COMMON/utilities.c add_path

We have /u/mdbs/UserFiles/COURSEsqlreq1

Enter r_read_file

Enter rd_temp_str_info

Exit3 rd_temp_str_info

Enter rd_temp_str_info

Exit3 rd_temp_str_info

Enter rd_temp_str_info

Exit1 rd_temp_str_info

Enter rd_temp_str_info

Exit3 rd_temp_str_info

Enter rd_temp_str_info

Exit3 rd_temp_str_info

Enter rd_temp_str_info

Exit1 rd_temp_str_info

Enter rd_temp_str_info

Exit3 rd_temp_str_info

Enter rd_temp_str_info

Exit3 rd_temp_str_info

Enter rd_temp_str_info

Exit1 rd_temp_str_info

Enter rd_temp_str_info

Exit3 rd_temp_str_info

Enter rd_temp_str_info

Exit3 rd_temp_str_info


```
Enter rd_temp_str_info
Exit3 rd_temp_str_info
Enter rd_temp_str_info
Exit1 rd_temp_str_info
Enter rd_temp_str_info
Exit3 rd_temp_str_info
Enter rd_temp_str_info
Exit3 rd_temp_str_info
Enter rd_temp_str_info
Exit3 rd_temp_str_info
Enter rd_temp_str_info
Exit1 rd_temp_str_info
Enter rd_temp_str_info
Exit3 rd_temp_str_info
Enter rd_temp_str_info
Exit3 rd_temp_str_info
Enter rd_temp_str_info
Exit3 rd_temp_str_info
Enter rd_temp_str_info
Exit3 rd_temp_str_info
Enter rd_temp_str_info
Exit3 rd_temp_str_info
Enter rd_temp_str_info
Exit3 rd_temp_str_info
Enter rd_temp_str_info
Exit3 rd_temp_str_info
Enter rd_temp_str_info
Exit3 rd_temp_str_info
Enter rd_temp_str_info
Exit3 rd_temp_str_info
Exit r_read_file
Exit r_read_transaction_file
Enter queries_to_KMS
Enter list_queries
```

```
1      select *
        from course
```

```
2      select *
      from fquarter

3      select *
      from wquarter

4      select *
      from course
      where prereq /= 'na'

5      select cnum,prereq
      from course
      where prereq /= 'na'
      or cnum = 'C100'
```

```
6      select cnum,title,dept,prereq
      from course
```

```
[7;7m-- more --[0;0m
```

```
      where cnum in
      (select cnum
      from wquarter)
```

```
7      select cnum,title,dept,prereq
      from course
      where cnum not in
      (select cnum
      from wquarter)
```

```
8      select
course.title,course.dept,wquarter.teacher
      from course,wquarter
      where course.cnum = wquarter.cnum
```

```
9      select cnum, prereq
      from course
      where cnum not in
            (select cnum
             from course
             where prereq = 'na')
```

Exit list_queries

Pick the number or letter of the action desired

- (num) - execute one of the preceding queries
- (d) - redisplay the file of queries
- (x) - return to the previous menu

[7;7mAction --- >[0;0m 1

```
Enter find_query
Exit find_query
SELECT * FROM COURSE
Enter sql_kernel_mapping_system
Enter rel_kms_info_alloc
Exit rel_kms_info_alloc
  Enter target_list_info_alloc
Exit target_list_info_alloc
  Enter set_kfs_ptr
set_kfs_ptr - ptr set
Exit set_kfs_ptr
Enter valid_relation
Exit valid_relation - COURSE rel found
FIRST ATTR: CNUM
  Enter r_insert_alias
Exit r_insert_alias
Enter r_copy_all_attr
```

```

Enter target_list_info_alloc
Exit target_list_info_alloc
Enter target_list_info_alloc
Exit target_list_info_alloc
Enter target_list_info_alloc
Exit target_list_info_alloc
Enter r_kms_info_cleanup
Exit r_kms_info_cleanup
Exit sql_kernel_mapping_system

```

CNUM	TITLE	DEPT	PREREQ
C100	Database	Cs	Na
C200	Proglang	Cs	Na
C300	Opersyst	Cs	C200
C305	Graphics	Cs	C200

Pick the number or letter of the action desired

- (num) - execute one of the preceding queries
- (d) - redisplay the file of queries
- (x) - return to the previous menu

```
[7;7mAction --- >[0;0m 2
```

```

Enter find_query
Exit find_query
SELECT * FROM FQUARTER
Enter sql_kernel_mapping_system
Enter rel_kms_info_alloc
Exit rel_kms_info_alloc
Enter abdl_tran_cleanup
Exit abdl_tran_cleanup
Enter target_list_info_alloc

```

```
Exit target_list_info_alloc
  Enter set_kfs_ptr
set_kfs_ptr - ptr set
Exit set_kfs_ptr
Enter valid_relation
Exit valid_relation - FQUARTER rel found
FIRST ATTR: CNUM
  Enter r_insert_alias
Exit r_insert_alias
Enter r_copy_all_attr
Enter target_list_info_alloc
Exit target_list_info_alloc
Enter r_kms_info_cleanup
Exit r_kms_info_cleanup
Exit sql_kernel_mapping_system
```

```
CNUM |TEACHER |
-----
```

```
C100 |Hsiao   |
```

```
C300 |Ritchie  |
```

Pick the number or letter of the action desired

(num) - execute one of the preceding queries

(d) - redisplay the file of queries

(x) - return to the previous menu

```
[7;7mAction --- >[0;0m 3
```

```
Enter find_query
```

```
Exit find_query
```

```
SELECT * FROM WQUARTER
```

```
Enter sql_kernel_mapping_system
```

```
Enter rel_kms_info_alloc
```

```

Exit rel_kms_info_alloc
Enter abdl_tran_cleanup
Exit abdl_tran_cleanup
  Enter target_list_info_alloc
Exit target_list_info_alloc
  Enter set_kfs_ptr
set_kfs_ptr - ptr set
Exit set_kfs_ptr
Enter valid_relation
Exit valid_relation - WQUARTER rel found
FIRST ATTR: CNUM
  Enter r_insert_alias
Exit r_insert_alias
Enter r_copy_all_attr
Enter target_list_info_alloc
Exit target_list_info_alloc
Enter r_kms_info_cleanup
Exit r_kms_info_cleanup
Exit sql_kernel_mapping_system

```

```

CNUM |TEACHER |
-----
C100 |Hsiao   |
C300 |Ritchie |

```

```

Pick the number or letter of the action desired
      (num) - execute one of the preceding queries
      (d)   - redisplay the file of queries
      (x)   - return to the previous menu

```

```
[7;7mAction --- >[0;0m 4
```

```
Enter find_query
```



```

Exit find_query
SELECT *          FROM COURSE          WHERE PREREQ /= 'NA'
Enter sql_kernel_mapping_system
Enter rel_kms_info_alloc
Exit rel_kms_info_alloc
Enter abdl_tran_cleanup
Exit abdl_tran_cleanup
  Enter target_list_info_alloc
Exit target_list_info_alloc
    Enter set_kfs_ptr
set_kfs_ptr - ptr set
Exit set_kfs_ptr
Enter valid_relation
Exit valid_relation - COURSE rel found
FIRST ATTR: CNUM
  Enter r_insert_alias
Exit r_insert_alias
Enter r_copy_all_attr
Enter target_list_info_alloc
Exit target_list_info_alloc
Enter target_list_info_alloc
Exit target_list_info_alloc
Enter target_list_info_alloc
Exit target_list_info_alloc
  Enter r_valid_attribute
rel: COURSE attr: PREREQ
Enter attr_function : PREREQ
Exit attr_function : Aggr: Attr: PREREQ
Exit r_valid_attribute - attr found
  Enter r_valid_attribute
rel: COURSE attr: PREREQ
Enter attr_function : PREREQ
Exit attr_function : Aggr: Attr: PREREQ
Exit r_valid_attribute - attr found

```

Enter r_kms_info_cleanup
Exit r_kms_info_cleanup
Exit sql_kernel_mapping_system

CNUM	TITLE	DEPT	PREREQ
C300	Opersyst	Cs	C200
C305	Graphics	Cs	C200

Pick the number or letter of the action desired
(num) - execute one of the preceding queries
(d) - redisplay the file of queries
(x) - return to the previous menu

[7;7mAction --- >[0;0m 5

Enter find_query
Exit find_query
SELECT CNUM, PREREQ FROM COURSE WHERE PREREQ /= 'NA'
OR CNUM = 'C100'
Enter sql_kernel_mapping_system
Enter rel_kms_info_alloc
Exit rel_kms_info_alloc
Enter abdl_tran_cleanup
Exit abdl_tran_cleanup
Enter target_list_info_alloc
Exit target_list_info_alloc
Enter target_list_info_alloc
Exit target_list_info_alloc
Enter set_kfs_ptr
set_kfs_ptr - ptr set
Exit set_kfs_ptr
Enter valid_relation

```
Exit valid_relation - COURSE rel found
FIRST ATTR: CNUM
    Enter r_insert_alias
Exit r_insert_alias
Enter r_valid_attribute
rel: COURSE attr: CNUM
Enter attr_function : CNUM
Exit attr_function : Aggr: Attr: CNUM
Exit r_valid_attribute - attr found
Enter r_valid_attribute
rel: attr: CNUM
Enter attr_function : CNUM
Exit attr_function : Aggr: Attr: CNUM
Exit r_valid_attribute - attr NOT found
Enter r_insert_alias
Exit r_insert_alias
Enter r_valid_attribute
rel: COURSE attr: PREREQ
Enter attr_function : PREREQ
Exit attr_function : Aggr: Attr: PREREQ
Exit r_valid_attribute - attr found
Enter r_valid_attribute
rel: attr: PREREQ
Enter attr_function : PREREQ
Exit attr_function : Aggr: Attr: PREREQ
Exit r_valid_attribute - attr NOT found
    Enter r_valid_attribute
rel: COURSE attr: PREREQ
Enter attr_function : PREREQ
Exit attr_function : Aggr: Attr: PREREQ
Exit r_valid_attribute - attr found
    Enter r_valid_attribute
rel: COURSE attr: PREREQ
Enter attr_function : PREREQ
```

```

Exit attr_function : Aggr: Attr: PREREQ
Exit r_valid_attribute - attr found
    Enter r_valid_attribute
rel: COURSE attr: CNUM
Enter attr_function : CNUM
Exit attr_function : Aggr: Attr: CNUM
Exit r_valid_attribute - attr found
    Enter r_valid_attribute
rel: COURSE attr: CNUM
Enter attr_function : CNUM
Exit attr_function : Aggr: Attr: CNUM
Exit r_valid_attribute - attr found
    Enter r_kms_info_cleanup
Exit r_kms_info_cleanup
Exit sql_kernel_mapping_system

```

```
CNUM | PREREQ |
```

```

-----
C100 | Na      |
C300 | C200    |
C305 | C200    |

```

Pick the number or letter of the action desired

- (num) - execute one of the preceding queries
- (d) - redisplay the file of queries
- (x) - return to the previous menu

```
[7;7mAction --- >[0;0m 6
```

```
Enter find_query
```

```
Exit find_query
```

```

SELECT CNUM, TITLE, DEPT, PREREQ FROM COURSE WHERE CNUM IN
(SELECT CNUM FROM WQUARTER)

```

Enter sql_kernel_mapping_system
Enter rel_kms_info_alloc
Exit rel_kms_info_alloc
Enter abdl_tran_cleanup
Exit abdl_tran_cleanup
Enter target_list_info_alloc
Exit target_list_info_alloc
Enter target_list_info_alloc
Exit target_list_info_alloc
Enter target_list_info_alloc
Exit target_list_info_alloc
Enter target_list_info_alloc
Exit target_list_info_alloc
Enter set_kfs_ptr
set_kfs_ptr - ptr set
Exit set_kfs_ptr
Enter valid_relation
Exit valid_relation - COURSE rel found
FIRST ATTR: CNUM
Enter r_insert_alias
Exit r_insert_alias
Enter r_valid_attribute
rel: COURSE attr: CNUM
Enter attr_function : CNUM
Exit attr_function : Aggr: Attr: CNUM
Exit r_valid_attribute - attr found
Enter r_valid_attribute
rel: attr: CNUM
Enter attr_function : CNUM
Exit attr_function : Aggr: Attr: CNUM
Exit r_valid_attribute - attr NOT found
Enter r_insert_alias
Exit r_insert_alias
Enter r_valid_attribute

```
rel: COURSE attr: TITLE
Enter attr_function : TITLE
Exit attr_function : Aggr: Attr: TITLE
Exit r_valid_attribute - attr found
Enter r_valid_attribute
rel: attr: TITLE
Enter attr_function : TITLE
Exit attr_function : Aggr: Attr: TITLE
Exit r_valid_attribute - attr NOT found
Enter r_insert_alias
Exit r_insert_alias
Enter r_valid_attribute
rel: COURSE attr: DEPT
Enter attr_function : DEPT
Exit attr_function : Aggr: Attr: DEPT
Exit r_valid_attribute - attr found
Enter r_valid_attribute
rel: attr: DEPT
Enter attr_function : DEPT
Exit attr_function : Aggr: Attr: DEPT
Exit r_valid_attribute - attr NOT found
Enter r_insert_alias
Exit r_insert_alias
Enter r_valid_attribute
rel: COURSE attr: PREREQ
Enter attr_function : PREREQ
Exit attr_function : Aggr: Attr: PREREQ
Exit r_valid_attribute - attr found
Enter r_valid_attribute
rel: attr: PREREQ
Enter attr_function : PREREQ
Exit attr_function : Aggr: Attr: PREREQ
Exit r_valid_attribute - attr NOT found
Enter r_valid_attribute
```

```

rel: COURSE attr: CNUM
Enter attr_function : CNUM
Exit attr_function : Aggr: Attr: CNUM
Exit r_valid_attribute - attr found
  Enter rel_kms_info_alloc
Exit rel_kms_info_alloc
  Enter target_list_info_alloc
Exit target_list_info_alloc
  Enter valid_relation
Exit valid_relation - WQUARTER rel found
FIRST ATTR: CNUM
Enter r_insert_alias
Exit r_insert_alias
Enter r_valid_attribute
rel: WQUARTER attr: CNUM
Enter attr_function : CNUM
Exit attr_function : Aggr: Attr: CNUM
Exit r_valid_attribute - attr found
Enter r_valid_attribute
rel: attr: CNUM
Enter attr_function : CNUM
Exit attr_function : Aggr: Attr: CNUM
Exit r_valid_attribute - attr NOT found
  Enter r_kms_info_cleanup
Exit r_kms_info_cleanup
Exit sql_kernel_mapping_system

```

CNUM	TITLE	DEPT	PREREQ
C100	Database	Cs	Na
C300	Opersyst	Cs	C200

Pick the number or letter of the action desired
 (num) - execute one of the preceding queries

- (d) - redisplay the file of queries
- (x) - return to the previous menu

[7;7mAction --- >[0;0m x

Exit queries_to_KMS
Enter r_free_requests
Exit r_free_requests

Enter mode of input desired

- (f) - read in a group of queries from a file
- (t) - read in queries from the terminal
- (m) - mass load a file
- (d) - display the current database schema
- (x) - return to the previous menu

[7;7mAction --- >[0;0m x

Exit r_process_old

Enter type of operation desired

- (l) - load new database
- (p) - process existing database
- (x) - return to the MLDS/MBDS system menu

[7;7mAction --- >[0;0m x

Enter r_save_catalog
Exit r_save_catalog
Exit r_language_interface_layer
Enter free_rel_user
Enter free_kfs_data
Exit free_kfs_data


```
Exit free_rel_user
exit sql main
```

The Multi-Lingual/Multi-Backend Database System

Select an operation:

- (a) - Execute the attribute-based/ABDL interface
- (r) - Execute the relational/SQL interface
- (h) - Execute the hierarchical/DL/I interface
- (n) - Execute the network/CODASYL interface
- (f) - Execute the functional/DAPLEX interface
- (o) - Execute the Object-Oriented interface
- (x) - Exit to the operating system

Select-> x

All done with MBDS

[7;7mdb11/u/mdbs/edwards/run--2>

APPENDIX B. EWIR DATABASE SUCCESSFUL LOAD SCRIPT

```
Script started on Tue Jun  4 11:10:41 1996
db11/u/mdbs/edwards/run> ebg
111213abdl.930809.txt          oldEWIRcat
EWIR_ABDL.r                    out*
Enter                          q4
Etest                          q4.2
Etest1                         sstop*
FAMILY.cover                   start.check
Family.cover.backup            start.cntrl*
MAINT.r                        start.cntrl.screen.trace*
WUcourse                       stop.check
WUtest                         stop.db11*
WUtest2                       stop.db12*
WUtest3                       stop.db13*
daplex.demo                    temp.txt
list2.stop                     tempjunk/
loadEWIR                       test4
loop_f                         trace/
main*                          typescript
master.run.be*                 z13*
max_n_cmds.cntrl              zero.db11*
memorytest                    zero.db12*
min_n_cmds.be*                zero.db13*
min_n_cmds.cntrl              zero.db13~*
old/
```

Welcome to MDDBS, today is Tue Jun 4 11:10:46 PDT 1996

Check the time each file was last compiled:

```

-rwxrwxr-x 1 mdba      262144 May 31  1995
../BE/NEWESTrecp.exe
-rwxrwxr-x 1 mdba      98304 May 31  1995 ../BE/bget.exe
-rwxrwxr-x 1 mdba      98304 May 31  1995 ../BE/bput.exe
-rwxrwxr-x 1 mdba     204800 May 31  1995 ../BE/cc.exe
-rwxrwxr-x 1 mdba      40960 May 31  1995 ../BE/dio.exe
-rwxrwxr-x 1 mdba     262144 Jan 22 13:39 ../BE/dirman.exe
-rwxrwxr-x 1 mdba     245760 Jan 22 13:39 ../BE/recp.exe

-rwxrwxr-x 1 mdba     106496 May 31  1995
../CNTRL/cget.exe
-rwxrwxr-x 1 mdba     106496 May 31  1995
../CNTRL/cput.exe
-rwxrwxr-x 1 mdba     114688 Jan 22 13:38 ../CNTRL/iig.exe
-rwxrwxr-x 1 mdba     966656 Jun  2  1995 ../CNTRL/o.exe
-rwxrwxr-x 1 mdba     966656 Jan 22 13:50
../CNTRL/old4ti.exe
-rwxrwxr-x 1 mdba     966656 Jan 12 11:47
../CNTRL/oldti2.exe
-rwxrwxr-x 1 mdba     966656 Jan 16 11:15
../CNTRL/oldti3.exe
-rwxrwxr-x 1 mdba     966656 Jan 31 00:34
../CNTRL/oldti4.exe
-rwxrwxr-x 1 mdba     966656 Feb 29 16:01
../CNTRL/oldti5.exe
-rwxrwxr-x 1 mdba     966656 Sep  4  1995
../CNTRL/oldtioriginal.exe
-rwxrwxr-x 1 mdba     114688 Jan 22 13:38 ../CNTRL/pp.exe
-rwxrwxr-x 1 mdba     180224 Jan 22 13:38
../CNTRL/reqp.exe
-rwxrwxr-x 1 mdba     966656 May 15 13:31 ../CNTRL/ti.exe

```

There should be 12 files listed, if not you need to

recompile.

Do you need to recompile any executable and/or copy the 6
executable files to each Back End
(bget, bput, cc, dio, dirman, recp.exe)? (y/n) n

The Current Configuration is:

Version Name: edwards

Controller: db11

1 Back End:
db13

WARNING: All data will be lost if you reconfigure

Do you wish to reconfigure the Back Ends? (y/n) n

Do you wish to use current database? (y/n) n

Zeroing backend meta disk on back end, db13...

File to zero = /dev/sd2c

Bytes to zero = 1000000

Bytes written...

102400

204800

307200

409600

512000

614400

716800

819200

921600

1000000

Zeroing backend data disk on back end, db13...

File to zero = /dev/sd4c

Bytes to zero = 716800

Bytes written...

102400

204800

307200

409600

512000

614400

716800

Removing CINBT and IIG AT tables on controller, db11...

Do you wish to run the Multi Modal, Multi Lingual,
Multi Backended Database System? (y/n) y

stopping processes on back end db13

no processes to kill on db13

stopping processes on db11, the controller

stop.db11: syntax error at line 13: '(' unexpected

EXECUTING: start.cntrl

starting 5 of 6 controller processes on db11...

EXECUTING: rsh db13 -n /u/mdbs/be.edwards/run.be &

EXECUTING: /u/mdbs/edwards/CNTRL/ti.exe 1

No match.

Running backend on db13...

[1] 9651

PID written to /u/mdbs/.ti.exe.pid

[2] 9652

[3] 9653

[4] 9654

[5] 9655

[6] 9656

**** Unlink error: No such file or directory

[0] 0000 System configured for 1 backend(s).

[H[2J

MBDS: Initializing communications...

Seconds remaining: 5 4 3 2 1 [H[2J The Multi-
Lingual/Multi-Backend Database System

Select an operation:

- (a) - Execute the attribute-based/ABDL interface
- (r) - Execute the relational/SQL interface
- (h) - Execute the hierarchical/DL/I interface
- (n) - Execute the network/CODASYL interface
- (f) - Execute the functional/DAPLEX interface
- (o) - Execute the Object-Oriented interface
- (x) - Exit to the operating system

Select-> r

Enter sql main

Enter new_rel_user

Exit new_rel_user

Enter r_language_interface_layer

Enter sql_init

Exit sql_init

Enter r_load_db_list

Exit r_load_db_list

Enter type of operation desired

- (l) - load new database
- (p) - process existing database
- (x) - return to the MLDS/MBDS system menu

[7;7mAction --- >[0;0m l

Enter r_load_new

[7;7m

Enter name of database ---->[0;0m EWIR

Enter mode of input desired

- (f) - read in a group of creates from a file
- (t) - read in creates from the terminal
- (x) - return to the main menu

[7;7mAction --- >[0;0m f

Enter r_read_transaction_file

[7;7m

What is the name of the CREATE/QUERY file ---->[0;0m
EWIRsqldb

We are in the COMMON/utilities.c add_path

We have /u/mdbs/UserFiles/EWIRsqldb

Enter r_read_file

Enter rd_temp_str_info

Exit3 rd_temp_str_info

Enter rd_temp_str_info

Exit3 rd_temp_str_info

Enter rd_temp_str_info

Exit3 rd_temp_str_info


```
Enter rd_temp_str_info
Exit2 rd_temp_str_info
Exit r_read_file
Exit r_read_transaction_file
Enter creates_to_KMS
Enter sql_kernel_mapping_system
  Enter valid_relation
Exit valid_relation - rel NOT found
  Enter insert_column_node
Exit insert_column_node
    Enter insert_column_node
Exit insert_column_node
    Enter insert_column_node
Exit insert_column_node
    Enter insert_column_node
Exit insert_column_node
    Enter insert_column_node
Exit insert_column_node
  Exit sql_kernel_mapping_system
Enter sql_kernel_mapping_system
  Enter valid_relation
Exit valid_relation - rel NOT found
  Enter insert_column_node
Exit insert_column_node
    Enter insert_column_node
Exit insert_column_node
    Enter insert_column_node
Exit insert_column_node
    Enter insert_column_node
Exit insert_column_node
    Enter insert_column_node
Exit insert_column_node
    Enter insert_column_node
Exit insert_column_node
    Enter insert_column_node
Exit insert_column_node
```

```
Enter insert_column_node
Exit insert_column_node
Enter insert_column_node
Exit insert_column_node
Exit sql_kernel_mapping_system
Enter sql_kernel_mapping_system
Enter valid_relation
Exit valid_relation - rel NOT found
Enter insert_column_node
Exit insert_column_node
Enter insert_column_node
Exit insert_column_node
Enter insert_column_node
Exit insert_column_node
Enter insert_column_node
Exit insert_column_node
Enter insert_column_node
Exit insert_column_node
Enter insert_column_node
Exit insert_column_node
Exit sql_kernel_mapping_system
Enter sql_kernel_mapping_system
Enter valid_relation
Exit valid_relation - rel NOT found
Enter insert_column_node
Exit insert_column_node
Enter insert_column_node
Exit insert_column_node
Enter insert_column_node
Exit insert_column_node
Enter insert_column_node
Exit insert_column_node
```

```
Enter insert_column_node
Exit insert_column_node
Enter insert_column_node
Exit insert_column_node
Enter insert_column_node
Exit insert_column_node
Exit sql_kernel_mapping_system
Enter sql_kernel_mapping_system
Enter valid_relation
Exit valid_relation - rel NOT found
Enter insert_column_node
Exit insert_column_node
Enter insert_column_node
Exit insert_column_node
Enter insert_column_node
Exit insert_column_node
Enter insert_column_node
Exit insert_column_node
Enter insert_column_node
Exit insert_column_node
Enter insert_column_node
Exit insert_column_node
Enter insert_column_node
Exit insert_column_node
Exit sql_kernel_mapping_system
Enter sql_kernel_mapping_system
Enter valid_relation
Exit valid_relation - rel NOT found
Enter insert_column_node
Exit insert_column_node
Enter insert_column_node
Exit insert_column_node
Enter insert_column_node
Exit insert_column_node
```

```
Enter insert_column_node
Exit insert_column_node
Enter insert_column_node
Exit insert_column_node
Enter insert_column_node
Exit insert_column_node
Exit sql_kernel_mapping_system
Enter sql_kernel_mapping_system
Enter valid_relation
Exit valid_relation - rel NOT found
Enter insert_column_node
Exit insert_column_node
Enter insert_column_node
Exit insert_column_node
Enter insert_column_node
Exit insert_column_node
Enter insert_column_node
Exit insert_column_node
Enter insert_column_node
Exit insert_column_node
Enter insert_column_node
Exit insert_column_node
Enter insert_column_node
Exit insert_column_node
Exit sql_kernel_mapping_system
Exit creates_to_KMS
Enter r_free_requests
Exit r_free_requests
We are in the COMMON/utilities.c add_path
We have /u/mdbs/UserFiles/EWIR.t
We are in the COMMON/utilities.c add_path
We have /u/mdbs/UserFiles/EWIR.d
```

Would you like to use the existing descriptor file, EWIR.d,
for indexing information?(y or n)

[7;7mAction --- >[0;0m n

[H[2J

The following are the Relations in the EWIR Database:

EMITTER
ANTENNA
DIRECTION
CIRCORELIP
MECHSCAN
MECHTRACK
SECTOR

Beginning with the first Relation, we will present each
Attribute of the relation. You will be prompted as to
whether
you wish to include that Attribute as an Indexing Attribute,
and, if so, whether it is to be indexed based on strict
EQUALITY, or based on a RANGE OF VALUES. If you do not want
to enter any indexes for your database, type an 'n' when
the Action --> prompt appears

Strike RETURN or 'n' when ready to continue.

[7;7mAction --- >[0;0m

[H[2JRelation name: EMITTER
Attribute Name: UNIQUEID

Do you wish to install this Attribute as an Indexing

Attribute?

- (n) - no; continue with next Attribute/Relation
- (e) - yes; establish this as an EQUALITY Attribute
- (r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m n

[H[2JRelation name: EMITTER
Attribute Name: WEAPONSYSTEM

Do you wish to install this Attribute as an Indexing
Attribute?

- (n) - no; continue with next Attribute/Relation
- (e) - yes; establish this as an EQUALITY Attribute
- (r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m n

[H[2JRelation name: EMITTER
Attribute Name: EMITFUNCTION

Do you wish to install this Attribute as an Indexing
Attribute?

- (n) - no; continue with next Attribute/Relation
- (e) - yes; establish this as an EQUALITY Attribute
- (r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m n

[H[2JRelation name: EMITTER
Attribute Name: EMITPTFGEN

Do you wish to install this Attribute as an Indexing
Attribute?

- (n) - no; continue with next Attribute/Relation
- (e) - yes; establish this as an EQUALITY Attribute
- (r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m n

[H[2JRelation name: EMITTER
Attribute Name: ERFECCM

Do you wish to install this Attribute as an Indexing
Attribute?

- (n) - no; continue with next Attribute/Relation
- (e) - yes; establish this as an EQUALITY Attribute
- (r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m n

[H[2JRelation name: ANTENNA
Attribute Name: ANTENNAID

Do you wish to install this Attribute as an Indexing
Attribute?

- (n) - no; continue with next Attribute/Relation

- (e) - yes; establish this as an EQUALITY Attribute
- (r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m n

[H[2JRelation name: ANTENNA
Attribute Name: ANTTYPE

Do you wish to install this Attribute as an Indexing
Attribute?

- (n) - no; continue with next Attribute/Relation
- (e) - yes; establish this as an EQUALITY Attribute
- (r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m n

[H[2JRelation name: ANTENNA
Attribute Name: ANTFUNCT

Do you wish to install this Attribute as an Indexing
Attribute?

- (n) - no; continue with next Attribute/Relation
- (e) - yes; establish this as an EQUALITY Attribute
- (r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m n

[H[2JRelation name: ANTENNA
Attribute Name: HORDIM

Do you wish to install this Attribute as an Indexing Attribute?

- (n) - no; continue with next Attribute/Relation
- (e) - yes; establish this as an EQUALITY Attribute
- (r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m n

[H[2JRelation name: ANTENNA
Attribute Name: VERTDIM

Do you wish to install this Attribute as an Indexing Attribute?

- (n) - no; continue with next Attribute/Relation
- (e) - yes; establish this as an EQUALITY Attribute
- (r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m n

[H[2JRelation name: ANTENNA
Attribute Name: ANTDIREC

Do you wish to install this Attribute as an Indexing Attribute?

- (n) - no; continue with next Attribute/Relation
- (e) - yes; establish this as an EQUALITY Attribute
- (r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m n

[H[2JRelation name: ANTENNA

Attribute Name: ACELPOL

Do you wish to install this Attribute as an Indexing
Attribute?

(n) - no; continue with next Attribute/Relation

(e) - yes; establish this as an EQUALITY Attribute

(r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m n

[H[2JRelation name: ANTENNA

Attribute Name: UNIQUEID

Do you wish to install this Attribute as an Indexing
Attribute?

(n) - no; continue with next Attribute/Relation

(e) - yes; establish this as an EQUALITY Attribute

(r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m n

[H[2JRelation name: DIRECTION

Attribute Name: RADPATID

Do you wish to install this Attribute as an Indexing
Attribute?

- (n) - no; continue with next Attribute/Relation
- (e) - yes; establish this as an EQUALITY Attribute
- (r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m n

[H[2JRelation name: DIRECTION
Attribute Name: ANTGAIN

Do you wish to install this Attribute as an Indexing
Attribute?

- (n) - no; continue with next Attribute/Relation
- (e) - yes; establish this as an EQUALITY Attribute
- (r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m n

[H[2JRelation name: DIRECTION
Attribute Name: SECCHAR

Do you wish to install this Attribute as an Indexing
Attribute?

- (n) - no; continue with next Attribute/Relation
- (e) - yes; establish this as an EQUALITY Attribute
- (r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m n

[H[2JRelation name: DIRECTION

Attribute Name: BWDTHAZ

Do you wish to install this Attribute as an Indexing Attribute?

- (n) - no; continue with next Attribute/Relation
- (e) - yes; establish this as an EQUALITY Attribute
- (r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m n

[H[2JRelation name: DIRECTION

Attribute Name: BWDTHEL

Do you wish to install this Attribute as an Indexing Attribute?

- (n) - no; continue with next Attribute/Relation
- (e) - yes; establish this as an EQUALITY Attribute
- (r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m n

[H[2JRelation name: DIRECTION

Attribute Name: FIRSTAZ

Do you wish to install this Attribute as an Indexing Attribute?

- (n) - no; continue with next Attribute/Relation
- (e) - yes; establish this as an EQUALITY Attribute

(r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m n

[H[2JRelation name: DIRECTION

Attribute Name: FIRSTEL

Do you wish to install this Attribute as an Indexing
Attribute?

(n) - no; continue with next Attribute/Relation

(e) - yes; establish this as an EQUALITY Attribute

(r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m n

[H[2JRelation name: CIRCORELIP

Attribute Name: POLARID

Do you wish to install this Attribute as an Indexing
Attribute?

(n) - no; continue with next Attribute/Relation

(e) - yes; establish this as an EQUALITY Attribute

(r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m n

[H[2JRelation name: CIRCORELIP

Attribute Name: POLARDATA

Do you wish to install this Attribute as an Indexing Attribute?

- (n) - no; continue with next Attribute/Relation
- (e) - yes; establish this as an EQUALITY Attribute
- (r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m n

[H[2JRelation name: CIRCORELIP
Attribute Name: SENSE

Do you wish to install this Attribute as an Indexing Attribute?

- (n) - no; continue with next Attribute/Relation
- (e) - yes; establish this as an EQUALITY Attribute
- (r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m n

[H[2JRelation name: CIRCORELIP
Attribute Name: AXRATIO

Do you wish to install this Attribute as an Indexing Attribute?

- (n) - no; continue with next Attribute/Relation
- (e) - yes; establish this as an EQUALITY Attribute
- (r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m n

[H[2JRelation name: CIRCORELIP

Attribute Name: COMMENT

Do you wish to install this Attribute as an Indexing
Attribute?

- (n) - no; continue with next Attribute/Relation
- (e) - yes; establish this as an EQUALITY Attribute
- (r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m n

[H[2JRelation name: CIRCORELIP

Attribute Name: CONTAGENCY

Do you wish to install this Attribute as an Indexing
Attribute?

- (n) - no; continue with next Attribute/Relation
- (e) - yes; establish this as an EQUALITY Attribute
- (r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m n

[H[2JRelation name: CIRCORELIP

Attribute Name: LASTUPDATE

Do you wish to install this Attribute as an Indexing
Attribute?

- (n) - no; continue with next Attribute/Relation
- (e) - yes; establish this as an EQUALITY Attribute
- (r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m n

[H[2JRelation name: MECHSCAN
Attribute Name: SCANID

Do you wish to install this Attribute as an Indexing
Attribute?

- (n) - no; continue with next Attribute/Relation
- (e) - yes; establish this as an EQUALITY Attribute
- (r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m n

[H[2JRelation name: MECHSCAN
Attribute Name: SMPAVGTIME

Do you wish to install this Attribute as an Indexing
Attribute?

- (n) - no; continue with next Attribute/Relation
- (e) - yes; establish this as an EQUALITY Attribute
- (r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m

Error - Invalid operation selected;

Please pick again

[7;7mAction --- >[0;0m n n

[H[2JRelation name: MECHSCAN

Attribute Name: THRESHOLDMEAS

Do you wish to install this Attribute as an Indexing
Attribute?

- (n) - no; continue with next Attribute/Relation
- (e) - yes; establish this as an EQUALITY Attribute
- (r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m n

[H[2JRelation name: MECHSCAN

Attribute Name: PLANSCAN

Do you wish to install this Attribute as an Indexing
Attribute?

- (n) - no; continue with next Attribute/Relation
- (e) - yes; establish this as an EQUALITY Attribute
- (r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m n

[H[2JRelation name: MECHSCAN

Attribute Name: STPSGABILITY

Do you wish to install this Attribute as an Indexing Attribute?

- (n) - no; continue with next Attribute/Relation
- (e) - yes; establish this as an EQUALITY Attribute
- (r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m n

[H[2JRelation name: MECHSCAN
Attribute Name: SCFUNCT

Do you wish to install this Attribute as an Indexing Attribute?

- (n) - no; continue with next Attribute/Relation
- (e) - yes; establish this as an EQUALITY Attribute
- (r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m n

[H[2JRelation name: MECHSCAN
Attribute Name: RADPATID

Do you wish to install this Attribute as an Indexing Attribute?

- (n) - no; continue with next Attribute/Relation
- (e) - yes; establish this as an EQUALITY Attribute
- (r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m n

[H[2JRelation name: MECHTRACK
Attribute Name: TRACKID

Do you wish to install this Attribute as an Indexing
Attribute?

- (n) - no; continue with next Attribute/Relation
- (e) - yes; establish this as an EQUALITY Attribute
- (r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m n

[H[2JRelation name: MECHTRACK
Attribute Name: PLANTRACK

Do you wish to install this Attribute as an Indexing
Attribute?

- (n) - no; continue with next Attribute/Relation
- (e) - yes; establish this as an EQUALITY Attribute
- (r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m n

[H[2JRelation name: MECHTRACK
Attribute Name: UMAXRA

Do you wish to install this Attribute as an Indexing
Attribute?

- (n) - no; continue with next Attribute/Relation
- (e) - yes; establish this as an EQUALITY Attribute
- (r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m n

[H[2JRelation name: MECHTRACK

Attribute Name: LMAXRA

Do you wish to install this Attribute as an Indexing Attribute?

- (n) - no; continue with next Attribute/Relation
- (e) - yes; establish this as an EQUALITY Attribute
- (r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m n

[H[2JRelation name: MECHTRACK

Attribute Name: UMAXRE

Do you wish to install this Attribute as an Indexing Attribute?

- (n) - no; continue with next Attribute/Relation
- (e) - yes; establish this as an EQUALITY Attribute
- (r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m n

[H[2JRelation name: MECHTRACK

Attribute Name: LMAXRE

Do you wish to install this Attribute as an Indexing Attribute?

- (n) - no; continue with next Attribute/Relation
- (e) - yes; establish this as an EQUALITY Attribute
- (r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m n

[H[2JRelation name: SECTOR
Attribute Name: SCANID

Do you wish to install this Attribute as an Indexing Attribute?

- (n) - no; continue with next Attribute/Relation
- (e) - yes; establish this as an EQUALITY Attribute
- (r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m n

[H[2JRelation name: SECTOR
Attribute Name: SECTORTYPE

Do you wish to install this Attribute as an Indexing Attribute?

- (n) - no; continue with next Attribute/Relation
- (e) - yes; establish this as an EQUALITY Attribute
- (r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m n

[H[2JRelation name: SECTOR
Attribute Name: UPPERLIMITS

Do you wish to install this Attribute as an Indexing
Attribute?

- (n) - no; continue with next Attribute/Relation
- (e) - yes; establish this as an EQUALITY Attribute
- (r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m n

[H[2JRelation name: SECTOR
Attribute Name: LOWERLIMITS

Do you wish to install this Attribute as an Indexing
Attribute?

- (n) - no; continue with next Attribute/Relation
- (e) - yes; establish this as an EQUALITY Attribute
- (r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m n

[H[2JRelation name: SECTOR
Attribute Name: SECWAZ

Do you wish to install this Attribute as an Indexing

Attribute?

- (n) - no; continue with next Attribute/Relation
- (e) - yes; establish this as an EQUALITY Attribute
- (r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m n

[H[2JRelation name: SECTOR
Attribute Name: SECWEL

Do you wish to install this Attribute as an Indexing
Attribute?

- (n) - no; continue with next Attribute/Relation
- (e) - yes; establish this as an EQUALITY Attribute
- (r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m n

[H[2JRelation name: SECTOR
Attribute Name: SMTRACK

Do you wish to install this Attribute as an Indexing
Attribute?

- (n) - no; continue with next Attribute/Relation
- (e) - yes; establish this as an EQUALITY Attribute
- (r) - yes; establish this as a RANGE Attribute

[7;7mAction --- >[0;0m n

Exit r_load_new

Enter type of operation desired

- (l) - load new database
- (p) - process existing database
- (x) - return to the MLDS/MBDS system menu

[7;7mAction --- >[0;0m x

Enter r_save_catalog

Exit r_save_catalog

Exit r_language_interface_layer

Enter free_rel_user

Enter free_kfs_data

Exit free_kfs_data

Exit free_rel_user

exit sql main

[H[2JThe Multi-Lingual/Multi-Backend Database System

Select an operation:

- (a) - Execute the attribute-based/ABDL interface
- (r) - Execute the relational/SQL interface
- (h) - Execute the hierarchical/DL/I interface
- (n) - Execute the network/CODASYL interface
- (f) - Execute the functional/DAPLEX interface
- (o) - Execute the Object-Oriented interface
- (x) - Exit to the operating system

Select-> x

All done with MBDS

[7;7mdb11/u/mdbs/edwards/run--2>[0;0m x

[7;7mdb11/u/mdbs/edwards/run--3>[0;0m

script done on Tue Jun 4 11:14:29 1996

script done on Tue Jun 4 11:14:29 1996

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center.....2
8725 John J. Kingman Rd. STE 0944
FT. Belvoir, VA 22060-6218
2. Dudley Knox Library.....2
Naval Postgraduate School
411 Dyer Rd
Monterey CA 93943-5101
3. Chairman, Computer Science Department..... 1
Naval Postgraduate School
Monterey, CA 93943-5000
4. Dr. C. Thomas Wu..... 3
Code CS/Wu
Computer Science Department
Naval Postgraduate School
Monterey, CA 93943-5000
5. Dr. David K. Hsiao.....1
Code CS/Hs
Computer Science Department
Naval Postgraduate School
Monterey, CA 93943-5000
6. Curricular Officer, Code 321
Naval Postgraduate School
Monterey, CA 93943-5000
7. LCDR Donna N. Scrivener, USN1
10 Rosemere Rd
Ballston Lake, NY 12019
8. LT Renell Edwards, USN.....1
Computer Science Department
Naval Postgraduate School
Monterey, CA 93943-5000
9. Sharon Cain1
NAIC/SCDD
4115 Hebble Creek Rd
Wright-Patterson AFB, OH 45433-5622
10. Doris Mleczek, Code P223051
Weapons Division
Naval Air Warfare Center
Pt. Mugu, CA 93042