HETEROGENEOUS DISTRIBUTED DATABASE MANAGEMENT: RIM-ORACLE INTERFACE

A Thesis

Presented to

the Faculty of the Department of Computer Science

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

By

Ngiam Woon Wong

December, 1987

ACKNOWLEDGEMENTS

The author is grateful to his advisor, Dr. Stephen Huang, for his help and guidance. He also wishes to thank his committee members, Dr. Ramez Elmasri and Dr. Tiee-Jian Wu, for their time and helpful suggestions.

He is deeply indebted to his parents for their love and support.

Finally, the author dedicates this thesis to Ms. Shirley Chee, for without her, this thesis would not be possible.

HETEROGENEOUS DISTRIBUTED DATABASE MANAGEMENT: RIM-ORACLE INTERFACE

An Abstract of a Thesis

Presented to

the Faculty of the Department of Computer Science

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

By

Ngiam Woon Wong

December, 1987

ABSTRACT

This thesis develops a database interface between two relational database management systems such that they can be viewed as a heterogeneous distributed database management system. The RIM-Oracle interface system allows a user to transparently access the Oracle DBMS through the RIM DBMS, where the two DBMS are located in different computers connected by a computer network.

The RIM-Oracle interface system allows a user to setup a data dictionary with information about existing relations stored in RIM and Oracle, and then perform RIM database retrieval and relational algebra queries on the relations. An interface program is build on top of each DBMS, and the two programs communicate over the network to perform their tasks. The host DBMS, RIM, is independent of the remote DBMS, Oracle, whereas the remote DBMS is dependent on the host DBMS, although the dependence is limited.

Since the host DBMS is independent of the remote DBMS, the interface system can be extended to allow queries involving other DBMSs, located in other nodes of the network, using the methods developed in this thesis.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

A database is a collection of related data stored in a computer, and a database management system (DBMS) is the software that is used to define, load and manipulate a database. A distributed database is a collection of related data stored in different computers connected by a computer network, and a distributed database management system (DDBMS) is the software that is used to define, load and manipulate a distributed database. A heterogeneous DDBMS is made up of different local DBMSs, whereas a homogeneous DDBMS is made up of similar local DBMSs [CERI 84, DATE 85, ULLMAN 82].

1.1 Thesis Description

RIM and Oracle are two relational DBMSs located in different computers connected by a computer network. This thesis develops a database interface between the two DBMSs such that they can be viewed as a heterogeneous distributed database management system. The RIM-Oracle interface system allows a user to transparently access the Oracle DBMS through the RIM DBMS. The RIM-Oracle interface system allows a user to setup a data dictionary with information

about existing relations stored in RIM and Oracle, and then perform RIM queries on the relations (Figure 1.1).

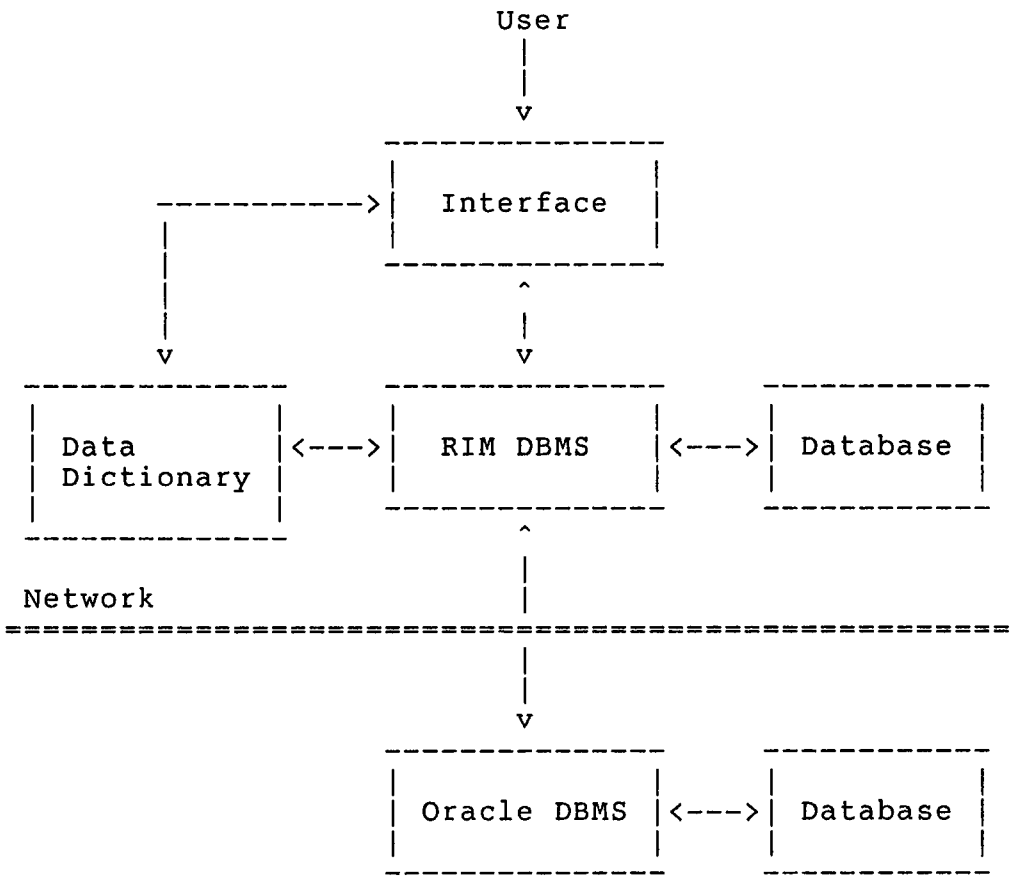


Figure 1.1 RIM-Oracle Interface System

A heterogeneous DDBMS is made up of different local DBMSs that must interface with each other to perform a query. Three approaches to building database interfaces between different DBMSs are of interest:

1. The first approach is to build an interface module for each pair of different DBMSs. If there are n different DBMSs, then O(n**2) interface modules are required.

2. The second approach is to build a common DBMS on top of the local DBMSs. This common DBMS is necessarily very complex, and requires a global data dictionary containing information about all databases in the DDBMS. But if there are n different DBMSs, then only n interface modules are required.

3. The third approach is to build a pair of interface modules for each different DBMS. Each local DBMS has a host and remote interface module, where the host module of one DBMS interfaces with the remote module of another DBMS, using a common query language. Each DBMS maintains a data dictionary containing information about all databases in the DDBMS. The host and remote modules are less

complex than the common DBMS of the second approach, and the remote module is less complex than the host module. If there are n different DBMSs, then 2n interface modules are required.

The third approach is actually a combination of the first and second approaches. If there are n different DBMSs, the first approach requires n-1 interfaces for each DBMS, and the second approach requires only 1 interface for each DBMS. The third approach requires 2 modules that are used to interface with other DBMSs. The second approach requires a common DBMS with its own query language to interface with local DBMSs. The third approach allows queries to be made to the local DBMSs in the query language of the local DBMSs - the interfaces communicate with each other using a common query language.

This thesis takes the third approach to building a heterogeneous DDBMS. A host interface module is implemented on top of RIM and a remote interface module is implemented on top of Oracle, and the common query language is RIM's query language.

1.2 Thesis Overview

This thesis consists of six chapters. Chapter 2 is a

discussion of the RIM DBMS, and Chapter 3 is a discussion of the Oracle DBMS. Chapter 4 presents the features of the RIM-Oracle interface system and Chapter 5 presents the design and implementation of the RIM-Oracle interface system. Finally, Chapter 6 concludes the thesis with a summary, a discussion of the complexity and performance of the system, and a suggestion of possible future extensions to the system.

Chapter 2

RIM DBMS

The Relational Information Management (RIM) system is a relational database management system originally developed for the National Aeronautics and Space Administration (NASA). RIM consists of two parts: (1) a standalone interactive system, and (2) a library of FORTRAN subroutines callable by application programs.

Two versions of RIM are used in this thesis, and both versions run on Digital Equipment Corporation's (DEC) VAX series of computers under the VMS operating system. The two versions are: (1) UHRIM (Version 1.0), from the University of Houston - Downtown [UHRIM 86], and (2) JPL RIM (Version 4.5), from the Jet Propulsion Laboratory, California Institute of Technology [JPLRIM 85].

The following is a description of the relevant modules of RIM (UH and JPL) to this thesis, using this notation:

lowercase characters	are to be supplied by the user
UPPERCASE characters	are keywords of the DBMS
[ ]	the enclosed items are optional
{ | }	choose one of the enclosed items separated by |

```
...                     the preceding item can be repeated
.                       not all the items are shown
.
.
```

2.1 Database Definition

Database definition involves defining the attributes and relations of the database. The database definition language of RIM is:

```
DEFINE database_name
OWNER  password
ATTRIBUTES
att_name type [length]
.
.
.
RELATIONS
rel_name WITH att_name_1 [att_name_2...att_name_n]
.
.
.
END
```

Three files (database_name1.DAT, database_name2.DAT and database_name3.DAT) are created in the current default directory to store the data dictionary and relations of the database. A correct password is required to define additional attributes and relations, and to query and

modify relations in the database. Attributes can be of type REAL for floating-point real numbers, INT for integer numbers, and TEXT for character strings. Relations will have attributes in the order they appear in the database definition statements.

2.2 Database Loading

Database definition is followed by database loading, either into a new relation or into a relation with data in it. Database loading in RIM is done by:

```
LOAD rel_name
value_1 value_2...value_n
.
.
.
END
```

Each tuple is loaded into the relation in a one-to-one correspondence with its attributes.

2.3 Query Language

After a RIM database has been defined, and its relations loaded with data, the database can be queried. The commands:

```
OPEN database_name
```

USER password

open an existing database with the correct password. RIM classifies queries into two categories: (1) Database Retrieval commands, and (2) Relational Algebra commands. The result of a database retrieval command is data from or about the database or relation; the result of a relational algebra command is a new relation.

2.3.1 Database Retrieval

Database retrieval commands are queries on the data dictionary or the relations of the database.

2.3.1.1 Listrel

The format of the LISTREL command is:

LISTREL { | rel_name | ALL }

The first form lists the relations defined in the database, the second form lists the definition of the relation, and the last form lists the definitions of all the relations in the database.

2.3.1.2 Exhibit

The format of the EXHIBIT command is:

EXHIBIT att_name_1 [att_name_2...att_name_n]

This command exhibits the relation or relations containing the set of attributes specified.

2.3.1.3 Select

The format of the SELECT command is:

```
SELECT    {att_name_1 [att_name_2...att_name_n] | ALL}
FROM       rel_name
[SORTED BY att_name...]
[WHERE     condition_1 [{AND | OR} condition_2...]]
```

This command selects all or part of the attributes of all the tuples from a relation satisfying the conditions specified, sorting the tuples by ascending order of the attributes specified.

Conditions are of the form:

att_name {EXISTS | FAILS}

att_name {EQ | NE | GT | GE | LT | LE} value

att_name_1 {EQA | NEA | GTA | GEA | LTA | LEA} att_name_2

where EQ = equal to

NE = not equal to

GT = greater than

GE = greater than or equal to

LT = less than

LE = less than or equal to

A = attribute

2.3.1.4 Tally

The format of the TALLY command is:

```
TALLY  att_name
FROM   rel_name
[WHERE condition_1 [{AND | OR} condition_2...]]
```

This command tallys the number of times each unique value of the attribute appears in the relation, according to the conditions specified.

2.3.1.5 Compute

The format of the COMPUTE command is:

```
COMPUTE {COUNT | MIN | MAX | AVE | SUM | SIGMA} att_name
FROM    rel_name
[WHERE  condition_1 [{AND | OR} condition_2...]]
```

This command computes the values of the attributes of the relation according to the function and conditions

specified.

2.3.2 Relational Algebra

Relational algebra commands are operations with one or two relations as operands, and a new relation as its result.

2.3.2.1 Project

The format of the PROJECT command is:

```
PROJECT  rel_name_1
FROM     rel_name_2
USING    {att_name_1 [att_name_2...att_name_n] | ALL}
[WHERE   condition_1 [{AND | OR} condition_2...]]
```

This command projects onto a new relation the tuples from an existing relation that satisfies the conditions specified, either with all the attributes or restricting the new relation to the attributes listed.

2.3.2.2 Join

The format of the JOIN command is:

```
JOIN     rel_name_1
USING    att_name_1
WITH     rel_name_2
```

```
USING   att_name_2
FORMING rel_name_3
[WHERE {EQ | NE | GT | GE | LT | LE}]
```

This command joins two relations to produce a third relation with all the attributes of the two relations. A tuple will exist in the new relation for every combination of tuples from relation 1 and 2, if the value of attribute 1 of a tuple from relation 1 and the value of attribute 2 of a tuple from relation 2 satisfy the condition specified.

2.3.2.3 Union

The format of the UNION command is:

```
UNION   rel_name_1
USING   att_name_1
WITH    rel_name_2
USING   att_name_2
FORMING rel_name_3
```

This command unions two relations to produce a third relation with all the attributes of the two relations. A tuple will exist in the new relation for every combination of tuples from relation 1 and 2, if the value of attribute 1 of a tuple from relation 1 and the value of attribute 2 of a tuple from relation 2 satisfy the equality condition.

Tuples that do not satisfy the condition are also included, with NULL values for the attributes that are missing.

2.3.2.4 Intersect

The format of the INTERSECT command is:

```
INTERSECT rel_name_1
WITH      rel_name_2
FORMING   rel_name_3
[USING    att_name_1 [att_name_2...att_name_n]]
```

This command intersects two relations to produce a third relation with the attributes specified, or if they are not specified, with all the attributes of the two relations, with duplicate attributes removed. A tuple will exist in the new relation if the values of the common attributes of a tuple from relation 1 are equal to the values of the common attributes of a tuple from relation 2.

2.3.2.5 Subtract

The format of the SUBTRACT command is:

```
SUBTRACT rel_name_1
FROM     rel_name_2
FORMING  rel_name_3
[USING   att_name_1 [att_name_2...att_name_n]]
```

This command subtracts relation 1 from relation 2 to produce a third relation with either all of the attributes of relation 2, or the restricted attributes from relation 2. A tuple from relation 2 will exist in the new relation if the values of its common attributes are not equal to the values of the common attributes of any tuple from relation 1.

2.4 Exit

The format of the EXIT command is:

EXIT

Finally, the EXIT command ends a session of RIM.

Chapter 3

ORACLE DBMS

Oracle is a relational database management system developed by the Oracle Corporation, with SQL as its database language. SQL can be used as: (1) a standalone interactive language, and as (2) a data sublanguage embedded in programming languages, such as FORTRAN, C and Pascal [ORACLE 81].

Oracle (Version 5.1.17) runs on Digital Equipment Corporation's VAX series of computers under the VMS operating system. SQL*Plus (Version 2.0.14), Oracle's implementation of SQL, is used in this thesis, and the following is a description of the relevant commands of SQL*Plus (using the notation of Chapter 2).

3.1 Data Definition

Data definition involves defining the tables and their attributes and lengths. (In SQL, relations are called tables.) The data definition command is:

```
CREATE TABLE table_name
            (att_name_1 type_1, att_name_2 type_2,...)
```

Attributes can be of type NUMBER(size, dec) for

floating-point real numbers, NUMBER(size) for integer numbers, and CHAR(size) for character strings, where size is the number of digits or characters in the attribute, and dec is the number of digits after the decimal point. Tables will have attributes in the order they appear in the data definition command.

3.2 Data Loading

Data definition is followed by data loading, either into a new table or into a table with data in it. Data loading in Oracle is done by:

```
INSERT INTO table_name
VALUES      (value_1, value_2,...,value_n)
```

This command inserts one tuple into the table, and each tuple is loaded in a one-to-one correspondence with the attributes of the table.

3.3 Query Language

After Oracle tables are defined and loaded with data, the tables can be queried.

3.3.1 Describe

The format of the DESCRIBE command is:

DESCRIBE table_name

This command describes the definition of the table, giving the type and length of each attribute of the table.

3.3.2 Select

The format of the SELECT command is:

```
SELECT att_name_1 [,att_name_2,...,att_name_n]
FROM   table_name
[WHERE condition_1 [{AND | OR} condition_2...]]
```

This command selects all or part of the attributes of all the tuples from a table satisfying the conditions satisfied.

Conditions are of the form:

```
att_name_1 {= | <> | > | >= | < | <=} {value | att_name_2}
```

where = is equal to
<> is not equal to
> is greater than
>= is greater than or equal to
< is less than
<= is less than or equal to

3.4 Exit

The format of the EXIT command is:

EXIT

Finally, the EXIT command ends a session of Oracle.

3.5 Comparison of RIM and Oracle

Although RIM and Oracle are relational DBMSs, the significant difference between the two systems is that the RIM query language is based on relational algebra, whereas the Oracle query language, SQL, is based on relational calculus.

The data types and precedence rules for executing a WHERE clause is RIM and Oracle are compared here:

1. Three data types of RIM and Oracle are of interest:

Data Type	RIM	Oracle
Integer	INT	NUMBER(size)
Real	REAL	NUMBER(size, dec)
Character	TEXT size	CHAR(size)

2. RIM executes a WHERE clause from left to right, with no precedence given to ANDs and ORs. SQL executes a WHERE clause by giving precedence to ANDs over ORs.

Chapter 4

RIM-ORACLE INTERFACE: SYSTEM FEATURES

The RIM-Oracle interface system allows a user to transparently access the Oracle DBMS through the RIM DBMS. A user can perform RIM queries, described in Chapter 2, on relations stored in both the RIM and Oracle DBMS.

The interface system is completely menu-driven, with different screens presenting menus from which the user is prompted to enter input, and to choose an operation to perform. The following presents the screens and menus, and describes the features and commands that are available in the interface system to a user.

4.1 Main Menu

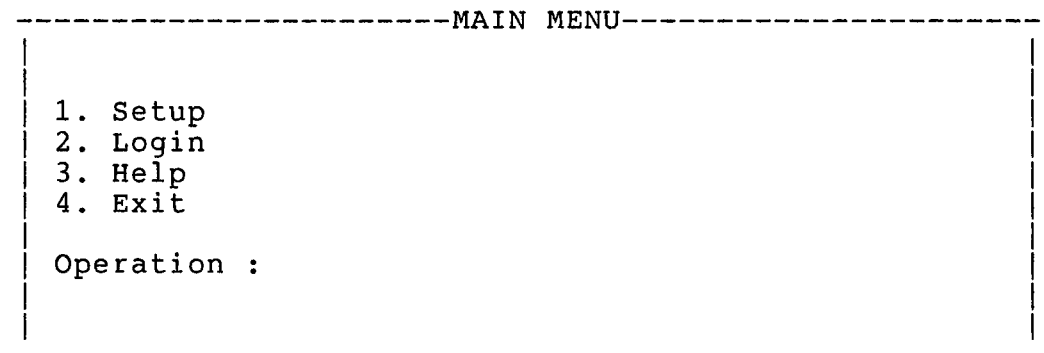


Figure 4.1 Main Menu Screen

This is the first screen presented to the user when he runs the interface system. The user may choose one of four of the above operations from the main menu of the interface system. Operation 1 allows a new user to setup the data dictionary required by the interface system. A user can create his own distributed database by setting up a data dictionary. Operation 2 allows an existing user to login to his own distributed database and perform queries on the relations there. Operation 3 allows the user to access an interactive help facility of help files with information on how to use the interface system. Finally, operation 4 allows the user to exit the system.

4.2 Login to Setup

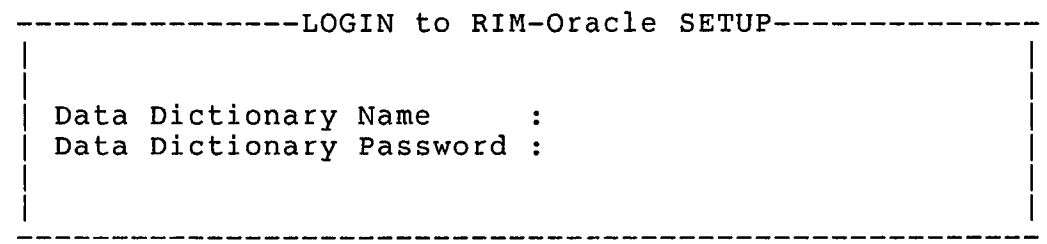


Figure 4.2 Login to RIM-Oracle Setup Screen

This operation allows a new user to setup the data dictionary required by the interface system. The data dictionary name and password supplied by the user will be used to create a database with two relations, TABLES and COLUMNS, containing all the information that is required by the interface system.

4.2.1 Setup

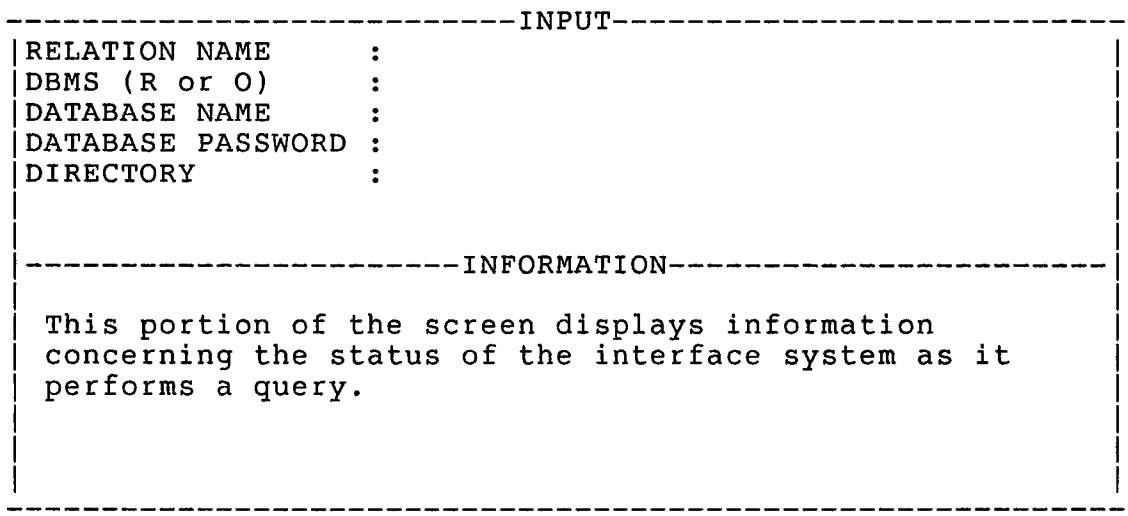


Figure 4.3 Setup Screen

This screen is presented after the user has provided the data dictionary name and password. An existing relation, in RIM or Oracle, the database name and password required to access the relation, and for RIM, the directory or location of the database files are required. This information is entered into the TABLES relation described earlier. The interface system also accesses the database specified to obtain information about the relation, namely its attributes' name, type and length. This information is entered into the COLUMNS relation described earlier.

4.3 Login

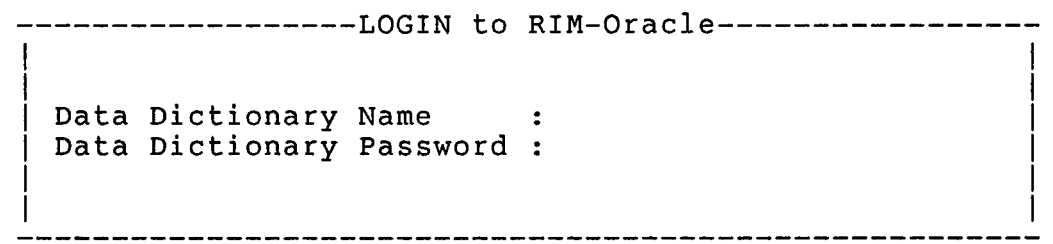


Figure 4.4 Login to RIM-Oracle Screen

This operation allows a user to login and access the interface system to perform queries on the relations in his own data dictionary. The data dictionary name and password supplied by the user must correctly identify a database created in the Setup operation for the user to access the next screen.

4.3.1 Menu

```
--------------------------MENU--------------------------
|                                                      |
|                                                      |
| 1. Database Retrieval                                |
| 2. Relational Algebra                                |
| 3. RIM                                               |
| 4. VAX/VMS                                           |
| 5. Setup                                             |
| 6. Main Menu                                         |
|                                                      |
| Operation :                                          |
|                                                      |
|                                                      |
--------------------------------------------------------
```

Figure 4.5 Menu Screen

This screen is presented after the user has provided the correct data dictionary name and password. Operation 1 gives the user access to database retrieval commands, and operation 2 gives the user access to relational algebra commands; both types of RIM query commands are described in Chapter 2 of this thesis.

If the user chooses operation 3, he will be prompted for a directory, which will become his current default directory from which he can access the RIM DBMS directly. IF the user chooses operation 4, he can access the VAX/VMS operating system directly. The EXIT command in RIM and the

LOGOUT command in VAX/VMS brings the user back to the interface system, to the above screen.

Operation 5 allows the user to access the Setup screen (Figure 4.3) to add information about relations into his data dictionary. Finally, operation 6 returns the user to the first screen (Figure 4.1), allowing the user to exit the system, or to login to or setup a different data dictionary.

4.3.2 Database Retrieval

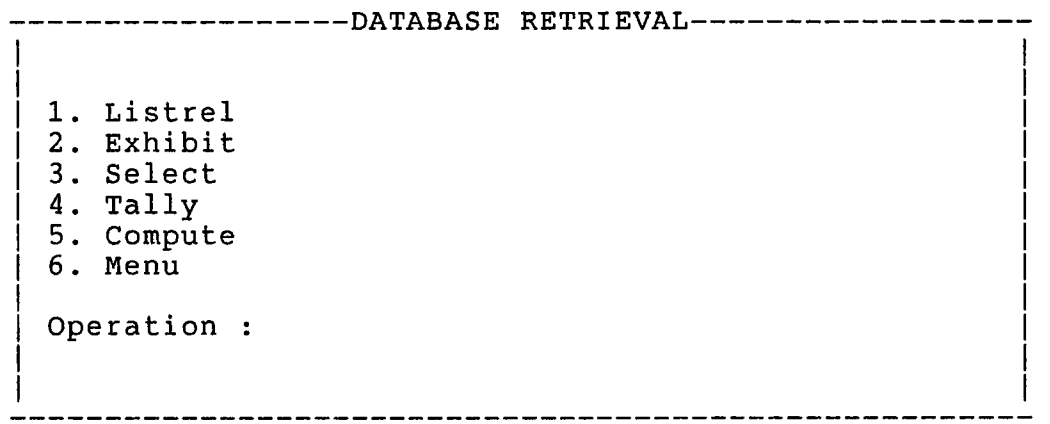


Figure 4.6 Database Retrieval Screen

This screen gives the user access to the above database retrieval commands, described in Chapter 2. The commands can successfully operate only if the user's data dictionary contains information about the relation being queried. Operation 6 returns the user to the MENU (Figure 4.5).

4.3.2.1 Listrel

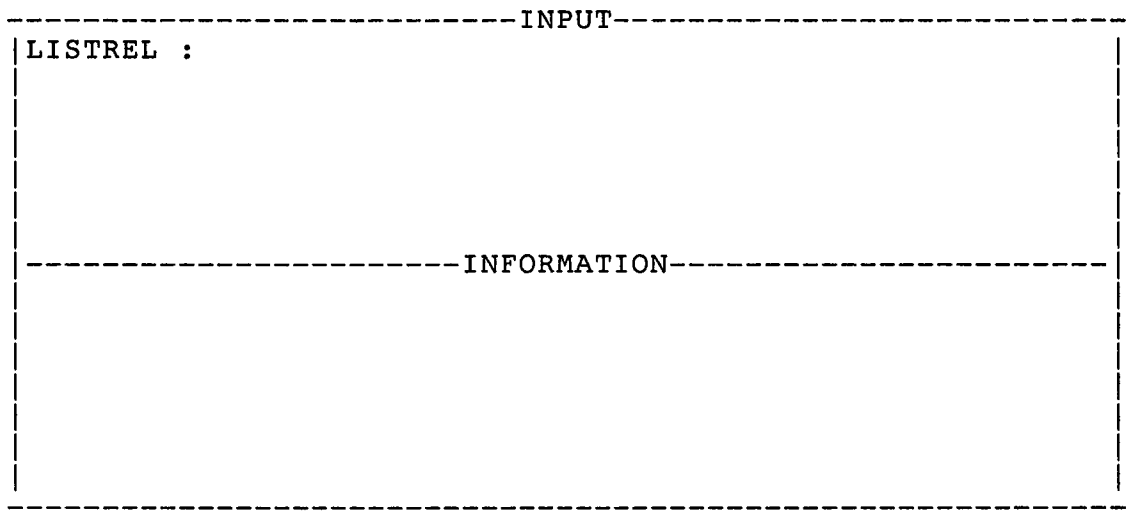


Figure 4.7 Listrel Screen

The LISTREL command (and the EXHIBIT command) does not query the relations or the databases in which they reside, but query the data dictionary setup by the user. The LISTREL command lists the relations in the data dictionary (TABLES), or lists the attributes' name, type and length of a specified relation, or of all the relations in the data dictionary (COLUMNS).

4.3.2.2 Exhibit

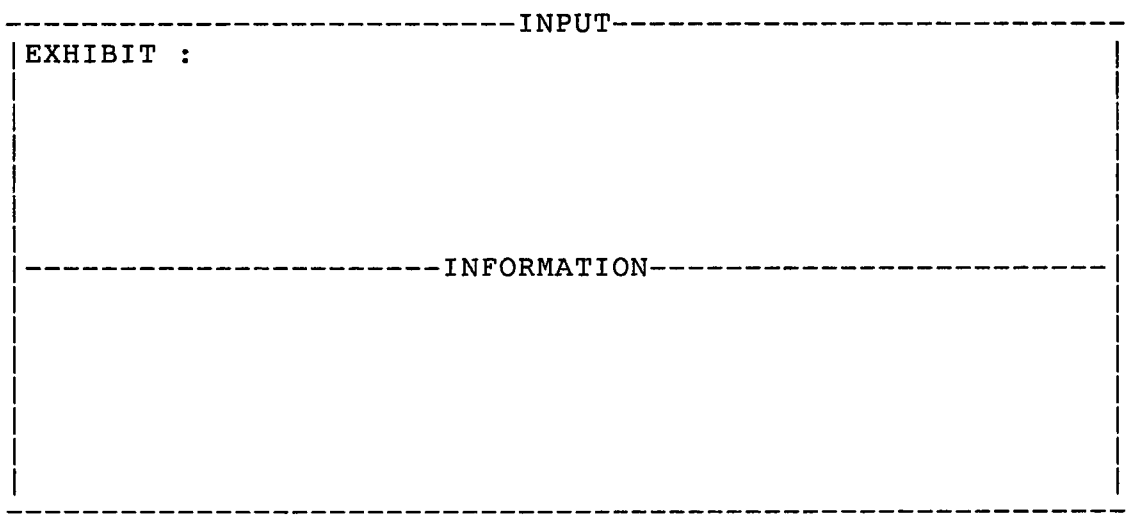


Figure 4.8 Exhibit Screen

The EXHIBIT command exhibits the relation or relations containing the set of attributes specified by querying the user's data dictionary (COLUMNS).

4.3.2.3 Select

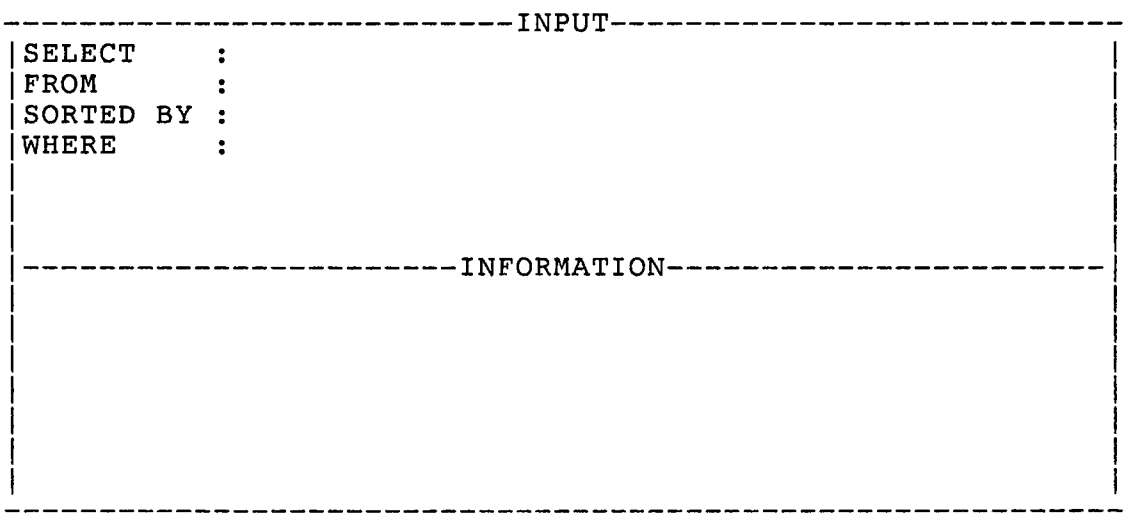


Figure 4.9 Select Screen

The SELECT command performs the select operation, either on a relation located in a RIM database, or in an Oracle database.

4.3.2.4 Tally

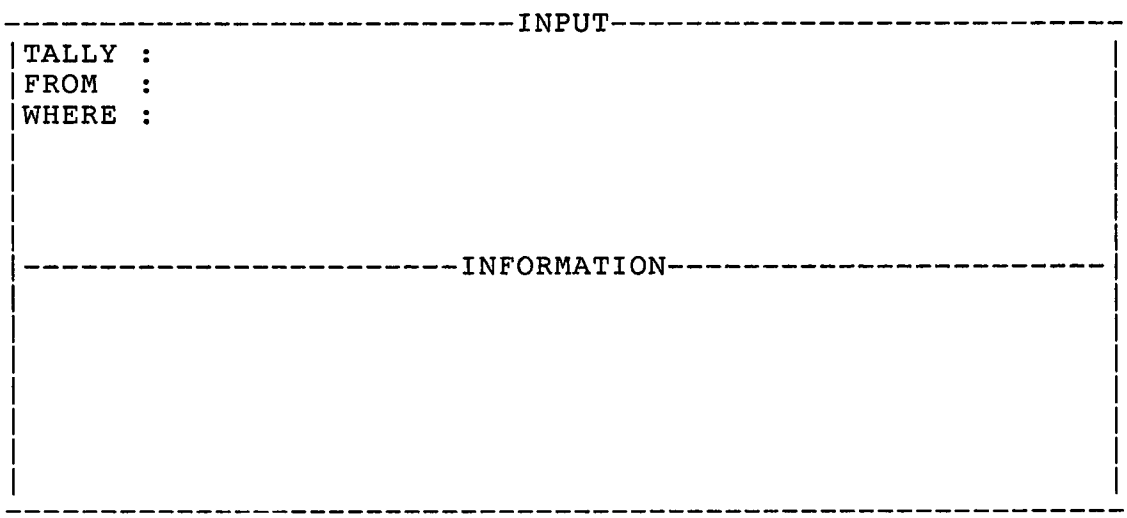


Figure 4.10 Tally Screen

The TALLY command performs the tally operation, either on a relation located in a RIM database, or in an Oracle database.

4.3.2.5 Compute

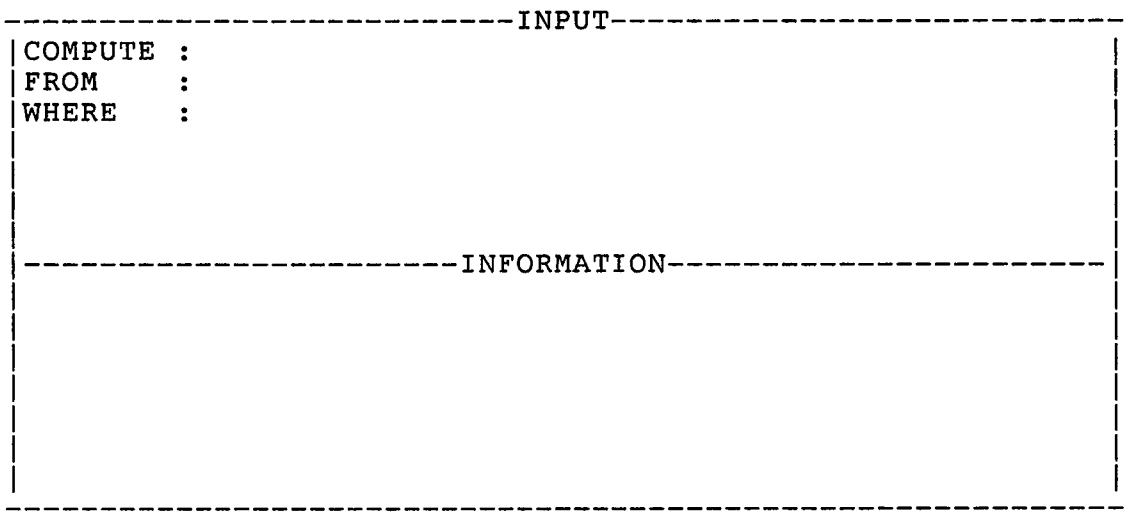


Figure 4.11 Compute Screen

The COMPUTE command performs the compute operation, either on a relation located in a RIM database, or in an Oracle database.

4.3.3 Relational Algebra

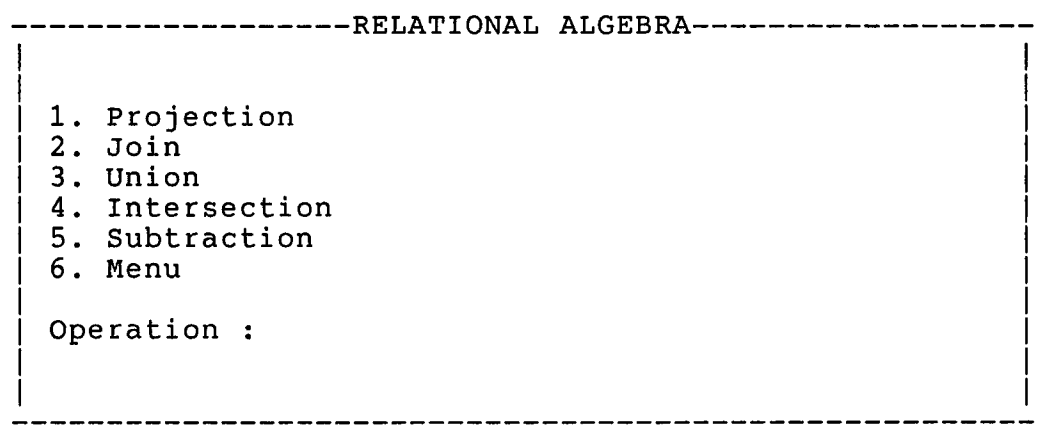


Figure 4.12 Relational Algebra Screen

This screen gives the user access to the above relational algebra commands, described in Chapter 2. The commands can successfully operate only if the user's data dictionary contains information about the relation being queried.

The result of a relational algebra command is a new relation. If one of the operand relations is located in a RIM database, the result relation is stored in that database. If not, the result relation is stored in a RIM database with the same name and password as the Oracle database of the operand relation. In either case,

information about the result relation is added to the user's data dictionary, and the new relation can thus be queried. Operation 6 returns the user to the MENU (Figure 4.5).

4.3.3.1 Project

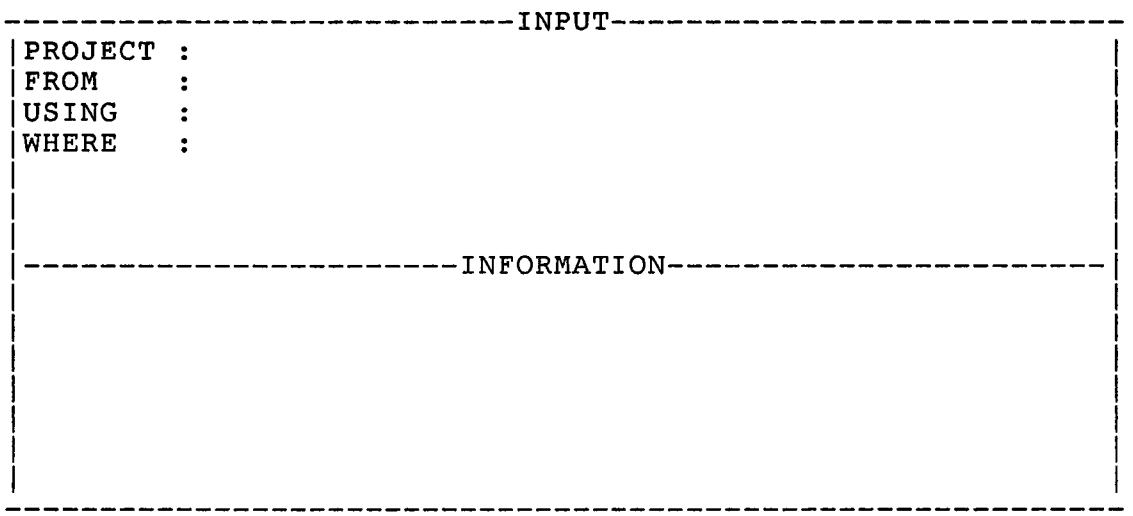


Figure 4.13 Project Screen

The PROJECT command performs the project operation, either on a relation located in a RIM database, or in an Oracle database.

4.3.3.2 Join

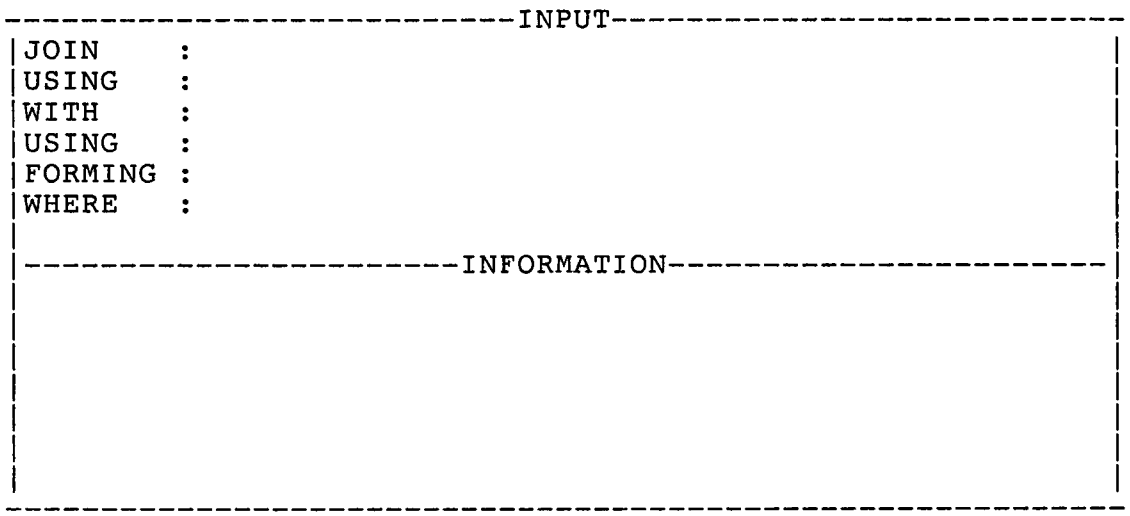


Figure 4.14 Join Screen

The JOIN command performs the join operation on two relations, and the two operand relations can be located in either a RIM or an Oracle database. The result relation is located in a RIM database.

4.3.3.3 Union

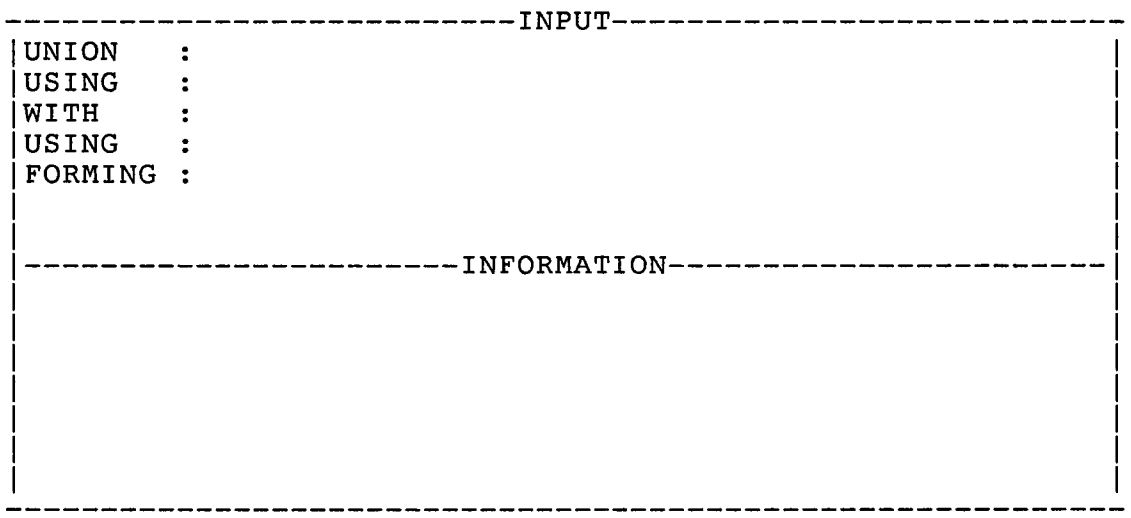


Figure 4.15 Union Screen

The UNION command performs the union operation on two relations, and the two operand relations can be located in either a RIM or an Oracle database. The result relation is located in a RIM database.

4.3.3.4 Intersect

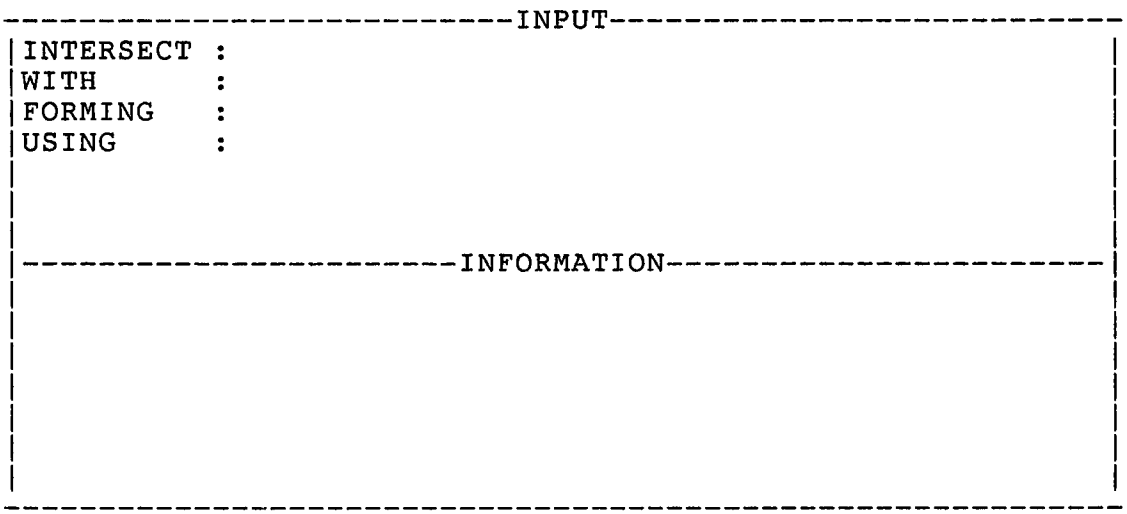


Figure 4.16 Intersect Screen

The INTERSECT command performs the intersect operation on two relations, and the two operand relations can be located in either a RIM or an Oracle database. The result relation is located in a RIM database.

4.3.3.5 Subtract

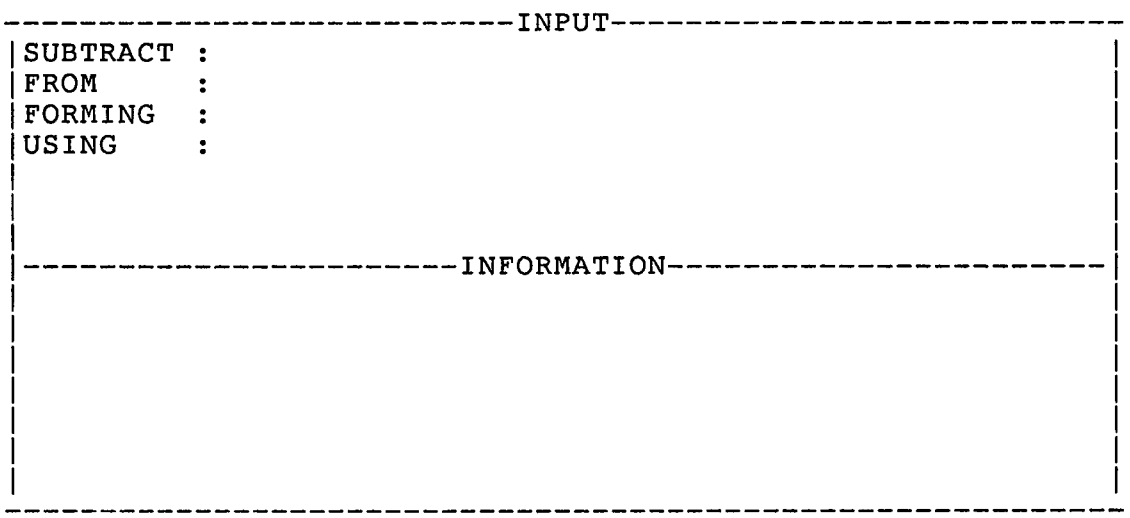


Figure 4.17 Subtract Screen

The SUBTRACT command performs the subtract operation on two relations, and the two operand relations can be located in either a RIM or an Oracle database. The result relation is located in a RIM database.

4.3.4 Output

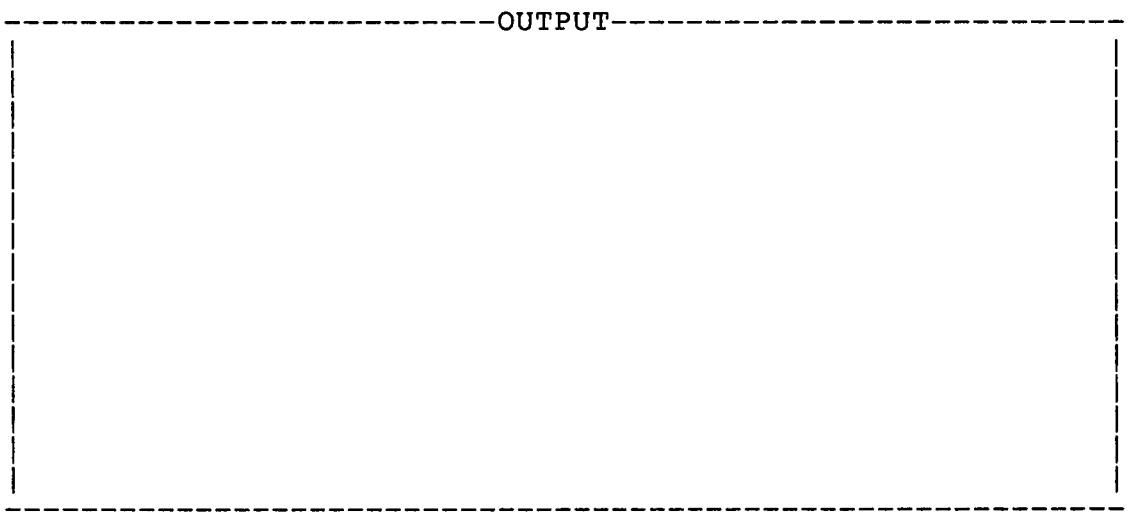


Figure 4.18 Output Screen

This screen is used by database retrieval and relational algebra commands to display the result of a query. The terminal width is set to 80 characters to display the result of a LISTREL or EXHIBIT command, and to 132 characters to display the result of all the other commands.

4.3.5 Error Messages

The following is a description of the error messages that may be displayed in the course of a session on the interface system. The format of each error message is:

Operation unsuccessful - error_message

where error_message can be one of the following:

(1) UHVAX2 unreachable
(2) Error in Data Dictionary Name/Password
(3) Error in Relation Name
(4) Error in Attribute Name
(5) Null Attribute/Relation Name not allowed
(6) Error in Function Name
(7) Null Input not allowed
(8) Error in Input
(9) Error in Relation/Database Name/Password

4.4 Help

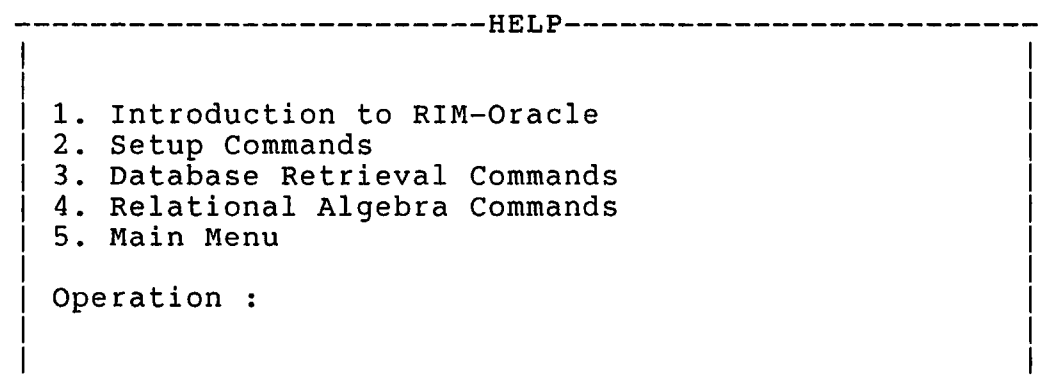


Figure 4.19 Help Screen

This operation allows the user to access an interactive help facility of help files with information on how to use the interface system. The help files are organized into the following topics: (1) Introduction to the RIM-Oracle interface system, (2) Setup commands, (3) Database Retrieval commands, and (4) Relational Algebra commands. The user chooses a topic, and the appropriate help file will be displayed with information on that topic. The user can return to the first screen (Figure 4.1) when he is ready to.

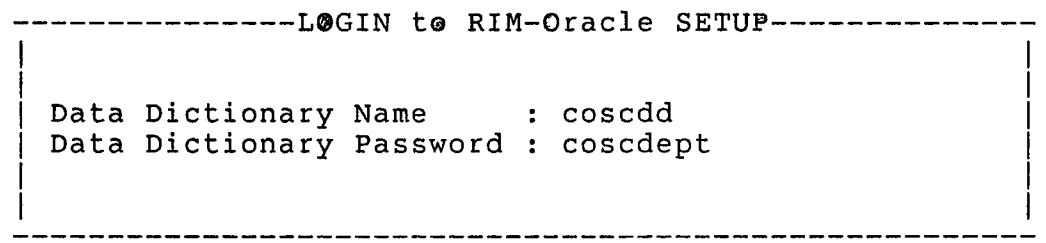
4.5 Exit

This operation allows the user to exit the interface system; this is the only exit in the system.

4.6 Example

The following is an example of a session on the interface system. A new user setups a distributed data dictionary of two relations - STUDENT in RIM and TRANSCPT in Oracle. The user then logins to his data dictionary, and performs queries on the relations. The data dictionary relations TABLES and COLUMNS are also shown, and are fully explained in Chapter 5.

The data dictionary name and password supplied by the user is used to create a database with two relations, TABLES and COLUMNS.

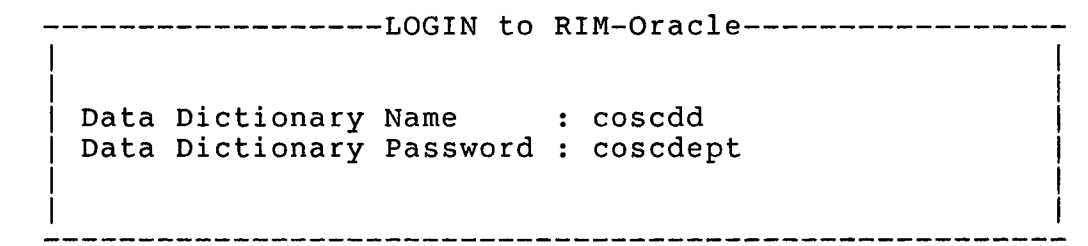


The user setups two relations in his data dictionary. Information about the two relations are entered into the data dictionary by the interface system:

```
----------------------------INPUT----------------------------
|RELATION NAME     : student                                 |
|DBMS (R or O)     : r                                       |
|DATABASE NAME     : coscdb                                  |
|DATABASE PASSWORD : coscdept                                |
|DIRECTORY         : [rim.rimdb.test]                        |
|                                                            |
|                                                            |
|-----------------------INFORMATION-------------------------|
|Accessing CSV1/RIM...                                       |
|  Query file created                                        |
|  Starting program in CSV1                                  |
|  STUDENT added to data dictionary                          |
|Operation successful                                        |
|                                                            |
|                                                            |
------------------------------------------------------------


----------------------------INPUT----------------------------
|RELATION NAME     : transcpt                                |
|DBMS (R or O)     : o                                       |
|DATABASE NAME     : uhdb                                    |
|DATABASE PASSWORD : uhadmin                                 |
|DIRECTORY         :                                         |
|                                                            |
|                                                            |
|-----------------------INFORMATION-------------------------|
|Accessing UHVAX2/Oracle...                                  |
|  Query file copied to UHVAX2                               |
|  Starting program in UHVAX2                                |
|  Result file received from UHVAX2                          |
|  TRANSCPT added to data dictionary                         |
|Operation successful                                        |
|                                                            |
------------------------------------------------------------
```

The user logins to his data dictionary, and performs queries on the relations in his data dictionary:



The LISTREL command lists the two relations defined in the data dictionary:

```
----------------------------INPUT----------------------------
|LISTREL :                                                   |
|                                                            |
|                                                            |
|                                                            |
|                                                            |
|                                                            |
|-----------------------INFORMATION-------------------------|
|Accessing CSV1/RIM...                                       |
|  Query file created                                        |
|  Starting program in CSV1                                  |
|Operation successful                                        |
|                                                            |
|                                                            |
|                                                            |
------------------------------------------------------------


--------------------------OUTPUT----------------------------
|EXISTING RELATIONS AS OF 20-NOV-1987 11:00:00.00            |
|                                                            |
| STUDENT                                                    |
| TRANSCPT                                                   |
|                                                            |
|                                                            |
|                                                            |
|                                                            |
|                                                            |
|                                                            |
|                                                            |
|                                                            |
|                                                            |
|                                                            |
------------------------------------------------------------
```

The user performs the SELECT command on the relations in his data dictionary:

```
----------------------------INPUT----------------------------
|SELECT    : all                                             |
|FROM      : student                                         |
|SORTED BY :                                                 |
|WHERE     :                                                 |
|                                                            |
|                                                            |
|                                                            |
|-----------------------INFORMATION-----------------------|
|Data Dictionary searched for STUDENT                        |
|  STUDENT is in CSV1/RIM                                    |
|Accessing CSV1/RIM...                                       |
|  Query file created                                        |
|  Starting program in CSV1                                  |
|Operation successful                                        |
|                                                            |
------------------------------------------------------------


----------------------------INPUT----------------------------
|SELECT    : all                                             |
|FROM      : transcpt                                        |
|SORTED BY :                                                 |
|WHERE     :                                                 |
|                                                            |
|                                                            |
|                                                            |
|-----------------------INFORMATION-----------------------|
|...                                                         |
|  Starting program in UHVAX2                                |
|  Result file received from UHVAX2                          |
|Accessing CSV1/RIM...                                       |
|  Query file created                                        |
|  Starting program in CSV1                                  |
|Operation successful                                        |
------------------------------------------------------------
```

The user joins the two relations in his data dictionary to create a new relation:

```
----------------------------INPUT----------------------------
|JOIN     : student                                          |
|USING    : ssn                                              |
|WITH     : transcpt                                         |
|USING    : ssn                                              |
|FORMING  : studtran                                         |
|WHERE    :                                                  |
|                                                            |
|-----------------------INFORMATION-----------------------   |
|...                                                         |
|  Result file received from UHVAX2                          |
|Accessing CSV1/RIM...                                       |
|  Query file created                                        |
|  Starting program in CSV1                                  |
|  STUDTRAN added to data dictionary                         |
|Operation successful                                        |
-------------------------------------------------------------
```

The LISTREL command lists the three relations defined in the data dictionary:

```
----------------------------INPUT-----------------------------
|LISTREL :                                                    |
|                                                             |
|                                                             |
|                                                             |
|                                                             |
|                                                             |
|-----------------------INFORMATION-------------------------|
|Accessing CSV1/RIM...                                        |
|  Query file created                                         |
|  Starting program in CSV1                                   |
|Operation successful                                         |
|                                                             |
|                                                             |
|                                                             |
-------------------------------------------------------------


---------------------------OUTPUT----------------------------
|EXISTING RELATIONS AS OF 20-NOV-1987 11:10:00.00             |
|                                                             |
| STUDENT                                                     |
| TRANSCPT                                                    |
| STUDTRAN                                                    |
|                                                             |
|                                                             |
|                                                             |
|                                                             |
|                                                             |
|                                                             |
|                                                             |
|                                                             |
|                                                             |
-------------------------------------------------------------
```

The data dictionary relations, TABLES and COLUMNS, are shown here, with information about the three relations.

Data Dictionary TABLES relation:

RNAME	DBMS	DBNAME	PWD	DIR
STUDENT	R	COSCDB	COSCDEPT	[RIM.RIMDB.TEST]
TRANSCPT	O	UHDB	UHADMIN	
STUDTRAN	R	COSCDB	COSCDEPT	[RIM.RIMDB.TEST]

Data Dictionary COLUMNS relation:

RNAME	ATT	TYPE	LEN
STUDENT	NAME	TEXT	4
STUDENT	SSN	TEXT	3
STUDENT	ADDR	TEXT	4
TRANSCPT	SECID	TEXT	4
TRANSCPT	SSN	TEXT	9
TRANSCPT	GRADE	TEXT	1
STUDTRAN	NAME	TEXT	4
STUDTRAN	SSN	TEXT	3
STUDTRAN	ADDR	TEXT	4
STUDTRAN	SECID	TEXT	1
STUDTRAN	SSN	TEXT	3
STUDTRAN	GRADE	TEXT	1

Chapter 5

RIM-ORACLE INTERFACE: DESIGN AND IMPLEMENTATION

The RIM DBMS is located in a DEC VAX-11/780 superminicomputer, under the VMS operating system (Version 4.4), of the Department of Computer Science, the University of Houston. The Oracle DBMS is located in a DEC VAX-11/785, under the VMS operating system (Version 4.5), of the University Computing Center, the University of Houston. The two computers are connected by DECnet, and their node names are CSV1 and UHVAX2 respectively.

The RIM-Oracle interface system allows a user to transparently access Oracle through RIM, and the design and implementation of this system is described in the following.

5.1 Overview

The RIM-Oracle interface system is executed in CSV1. A user must first setup a data dictionary with all the information required before he can perform queries on the relations in his data dictionary. The relations can be located in RIM databases on CSV1, or in Oracle databases on UHVAX2 (Figure 5.1).

Except for the LISTREL and EXHIBIT commands, which involve the data dictionary, all other database retrieval and relational algebra commands involve relations located in either a RIM or an Oracle database. If one relation is queried, it can be located in a RIM or an Oracle database. If two relations are queried, they can both be located in a RIM database, they can both be located in an Oracle database, or they can be located in a RIM database and an Oracle database.

The result of a relational algebra command is a new relation. If one of the operand relations is located in a RIM database, the result relation is stored in that database. If not, the result relation is stored in a RIM database with the same name and password as the Oracle database of the operand relation. In either case, information about the result relation is added to the data dictionary, and the new relation can thus be queried.

When a query is made to the interface system, the data dictionary is searched for information concerning the relations involved. If the relations are located in a RIM database, the query can be performed directly since the query is already in RIM's syntax. If a relation is located in an Oracle database, a query file is generated and copied

to the remote node. This query file contains an Oracle database name and password obtained from the distributed data dictionary, and a RIM query on the relation involved. The remote node program is activated, and takes the query file as input, producing another query file containing the Oracle database name and password, and an Oracle query. This file is then executed by Oracle, and a file containing the result of the query on the Oracle database specified is produced. This result file is in Oracle format and must be converted into the format expected by the host node program.

The result file is copied to the host node, where the host node program will load the result of the query on the Oracle database into the RIM database. (This process can be repeated if both relations are located in an Oracle database.) When both relations are located in a RIM database, the original query from the user can be performed, and the result displayed to the user.

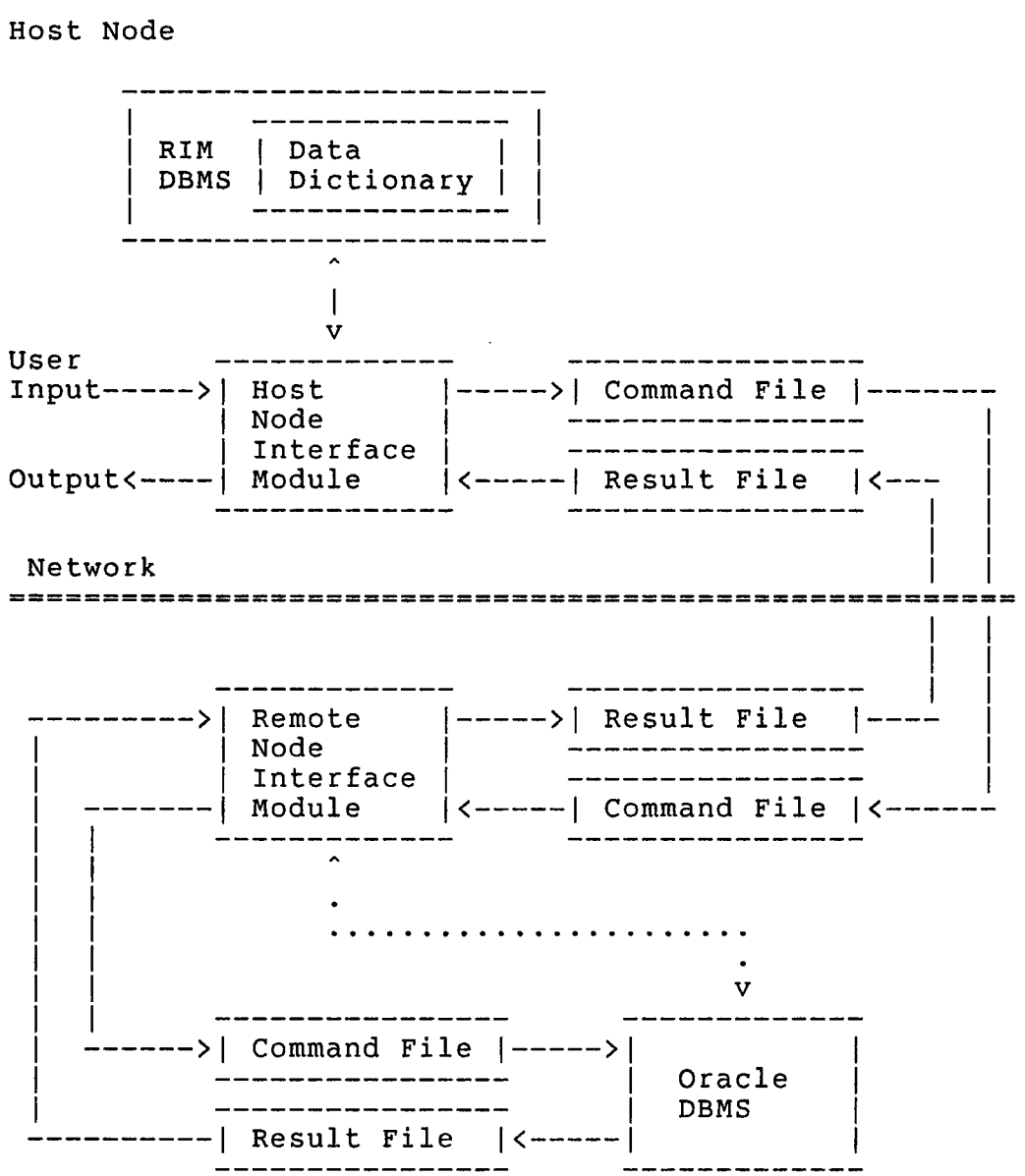


Figure 5.1 RIM-Oracle Interface System

5.2 Directory Structure

A fixed directory structure is imposed in both the host node and remote node by the interface system. A user must create this directory structure in his accounts on both nodes to successfully run the interface system.

5.2.1 Host Node Directory Structure

The host node directory structure consists of five bottom-level subdirectories located under one top-level subdirectory (Figure 5.2). The top-level subdirectory, RIMDB, can be located anywhere in the user's account, and can be of another name. The bottom-level subdirectories must be all located under the top-level subdirectory, and must be of the names specified. Subdirectory DD contains the user's data dictionary; subdirectory PGM contains the modules, programs and command procedures which make up the interface system on the host node, and is the current default directory of the user when he runs the interface system; subdirectory RES contains intermediate, temporary and result files produced by the interface system; subdirectory TEST contains the RIM database that is created with the same name and password as the Oracle database when all relations in a query involve Oracle; and subdirectory HELP contains the help files of the interface system.

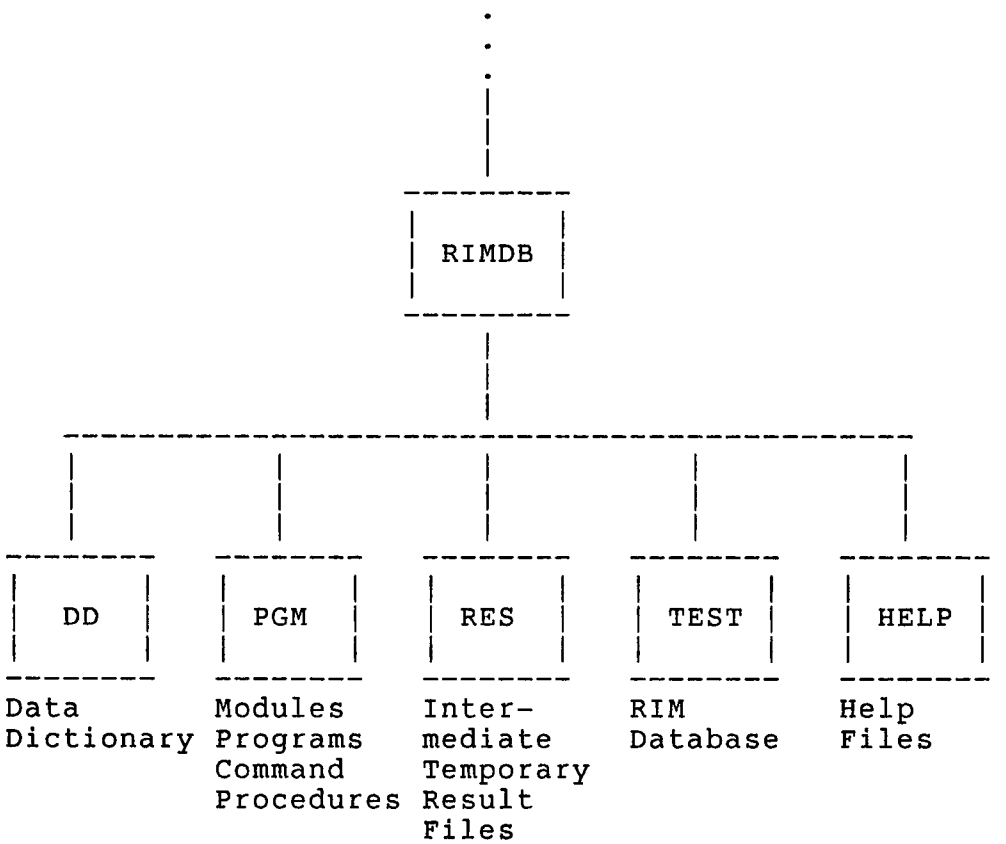


Figure 5.2 Host Node Directory Structure

5.2.2 Remote Node Directory Structure

The remote node directory structure consists of two bottom-level subdirectories located under one top-level subdirectory (Figure 5.3). The top-level subdirectory, Oracle, can be located anywhere in the user's account, and can be of another name. The bottom-level subdirectories must be both located under the top-level subdirectory, and must be of the names specified. Subdirectory PGM contains the modules, programs and command procedures which make up the interface system on the remote node; and subdirectory RES contains intermediate, temporary and result files produced by the interface system.

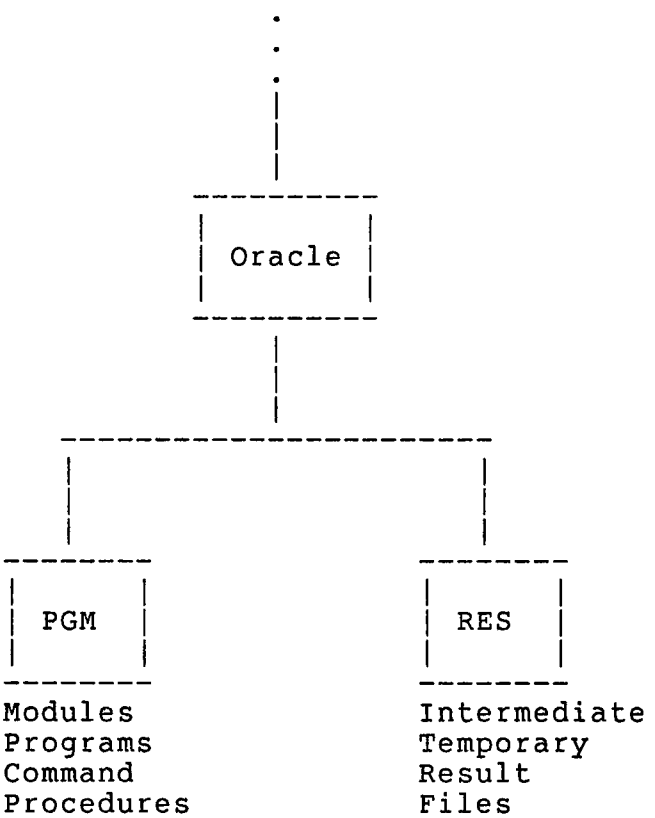


Figure 5.3 Remote Node Directory Structure

5.3 Data Dictionary

A data dictionary must be setup by a user using the SETUP commands described in Chapter 4. This data dictionary is actually a RIM database with two relations, TABLES and COLUMNS, created with information provided by the user, and the system catalog of both DBMSs (Figure 5.4).

The user provides the name and password of the data dictionary, and the relation names, in RIM or Oracle, the database names and passwords required to access the relations, and for RIM, the directories or locations of the database files containing the relations. The interface system also accesses the databases specified to obtain information about the relations, namely its attributes' name, type and length.

This information is used to create a database in RIM contained in three files (database_name1.DAT, database_name2.DAT and database_name3.DAT) located in the DD subdirectory. The TABLES relation has attributes RNAME (for the relation name), DBMS (RIM or Oracle), DBNAME (database name), PWD (database password), and DIR (directory). The COLUMNS relation has attributes RNAME, ATT (for attribute name), TYPE (REAL, INT or TEXT), and LEN

(length of attribute).

After a data dictionary has been created, a user can then login to and access the interface system to perform queries on the relations in his data dictionary. He can also access the data dictionary directly through RIM, if so desired.

RNAME	DBMS	DBNAME	PWD	DIR
Relation Name	RIM or Oracle	Database Name	Database Password	Directory

TABLES relation

RNAME	ATT	TYPE	LEN
Relation Name	Attribute Name	REAL, INT or TEXT	Length

COLUMNS relation

Figure 5.4 Data Dictionary with TABLES and COLUMNS Relations

The data dictionary is represented internally by the interface system in the form of linked lists (Figure 5.5). The interface system accesses the data dictionary to setup this data structure when the user logins, creating the TABLES linked list, and when he performs queries, creating the COLUMNS linked list for the relations specified. Information about new relations created by queries are added to the data dictionary, and to the TABLES linked list, so that they can also be queried.

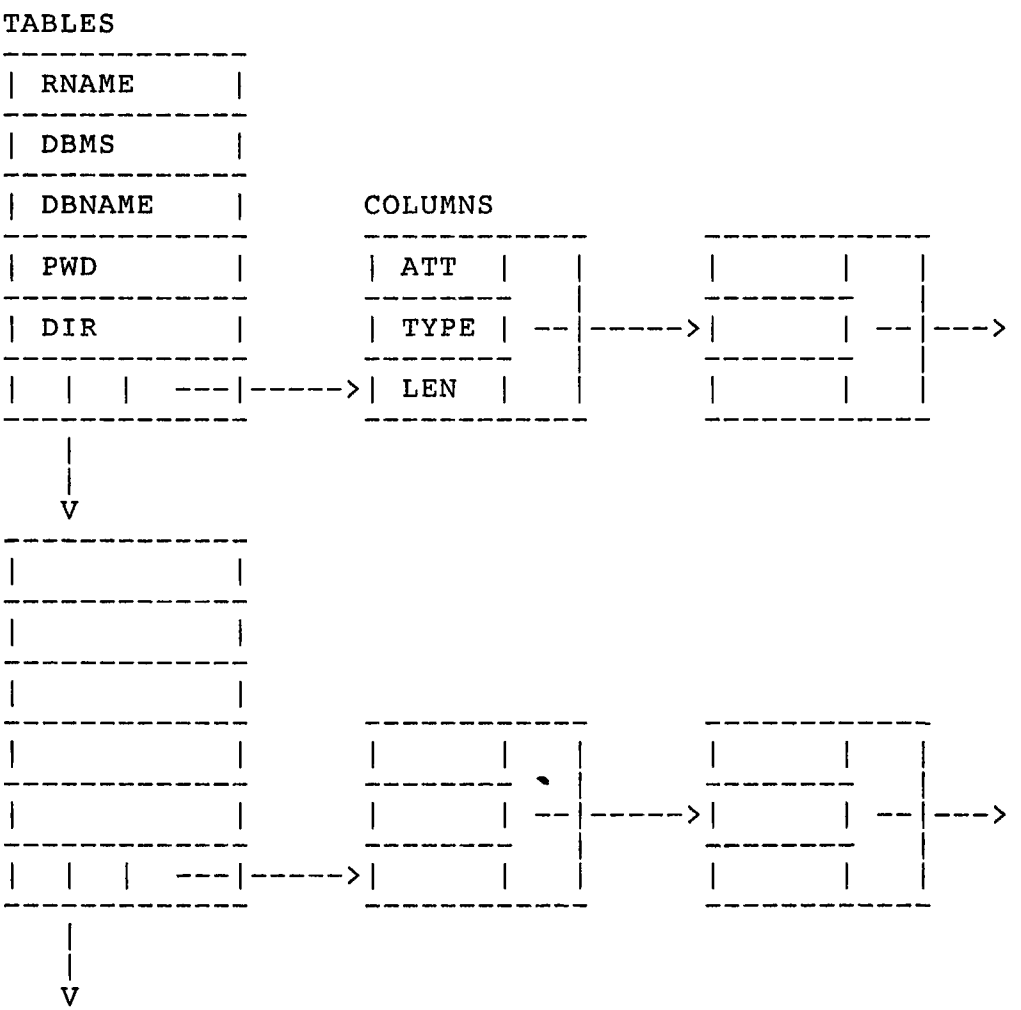


Figure 5.5 Data Structure of TABLES and COLUMNS Linked Lists

5.4 DECnet

The host node program activates the remote node program when a query involves a relation stored in Oracle. This is performed using DECnet-VAX transparent task-to-task communication between two tasks, or programs, located in different nodes of a network [VMS 86(B)]. The two remote Pascal programs communicate by using the same commands as for local files, namely with input/output procedures. The following is a description of the method used by the host node program to communicate with the remote node program:

1. The host node program first requests a logical connection with the remote node program by an OPEN statement specifying a remote task. It then waits for a termination message from the remote node program, informing the host node program that the remote node program has finished execution.

2. The task, a command procedure located in the default login directory of the account of the remote node specified, is started by the task-to-task connection request, and executes the remote node program.

3. The remote node program accepts the connection by an OPEN statement, and performs its tasks. On completion, it writes a termination message to inform the host node program that it has finished execution.

4. On receiving the termination message, the host node program can continue processing.

The following are the relevant portions of the host node and remote node programs described above:

Host node program:

```
OPEN(FILE_VARIABLE := out_file,
     FILE_NAME     := 'node"username password"::"TASK=test"',
     HISTORY       := NEW);

RESET(out_file);                          { Request connection,  }
READ(out_file, i);                        { and wait for         }
CLOSE(out_file);                          { termination message. }
```

test.COM:

```
$ set default [.ORACLE.PGM]  { Command procedure started    }
$ run Oracle_RIM.exe         { by connection request;       }
$ exit                       { executes remote node program.}
```

Remote node program:

```
OPEN(FILE_VARIABLE := in_file,        { Accept  connection,  }
     FILE_NAME     := 'SYS$NET',
     HISTORY       := OLD);

.
.
.

REWRITE(in_file);
WRITE(in_file, i);                        { and send             }
CLOSE(in_file);                           { termination message. }
```

5.5 System Implementation

The RIM-Oracle interface system is developed in Pascal, as implemented by DEC for VAX/VMS (VAX Pascal Version 3.5). VAX Pascal modules are used in the implementation to facilitate coding, and for future maintenance and modification [VMS 85].

5.5.1 Host Node Modules and Program

Eight modules and one main program form the host node portion of the interface system (Figure 5.6).

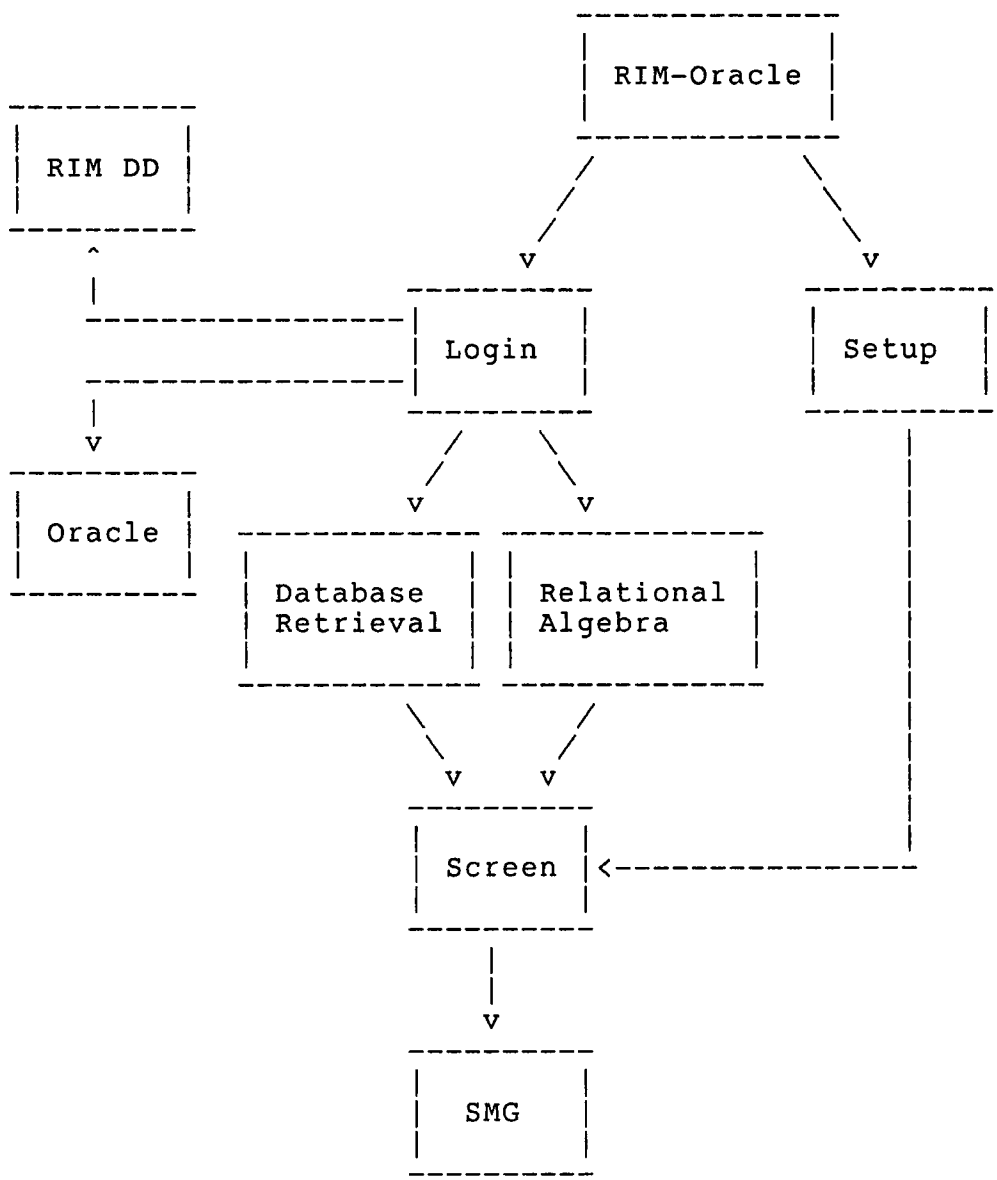


Figure 5.6 Host Node Module Structure

5.5.1.1 SMG Module

This module contains the declarations of external VAX/VMS Run-Time Library Screen Management procedures required for the screen manipulations described in Chapter 4 [VMS 86(C)]. The Screen Management Facility provides composition aids, in the form of terminal independent procedures, that are used by the menu-driven interface program.

5.5.1.2 Screen Module

This module contains the procedures for creating the screens described in Chapter 4. These procedures call on the external procedures of the SMG module. This module also contains procedures for displaying the output of a query, and procedures for displaying error messages associated with a query.

5.5.1.3 RIM DD Module

This module contains the procedures for extracting information from the data dictionary relations TABLES and COLUMNS, creating the linked lists described earlier. When a user logins to the program, the TABLES linked list is created. When the user first queries a relation, the COLUMNS linked list is created for that relation.

The following is a description of the method used to access a database to retrieve data from it; this method is used to access RIM (and Oracle) by other modules also:

1. A query file is created with commands to open a database, and to query the relations in it.

2. A subprocess is started by the program that will access the DBMS using the query file as input, and producing as output the result of the query [VMS 86(D)].

3. Once the subprocess is started, the program places itself into hibernation. When the subprocess is done, it wakes up the program.

4. The program then accesses the result file which contains the result of the query, either retrieving data from it, or displaying the data to the user.

5.5.1.4 Oracle Module

This module contains the procedures for creating a query file and copying the query file to the remote node, and for activating the remote node program. This module also adds information about new relations created by a

relational algebra command into the data dictionary, and to the TABLES linked list.

The following is a description of the method used to access Oracle in the remote node when a query involves relations stored there:

1. A query file is created in this format:

```
1
Oracle_database_name
Oracle_database_password
{att_name_1 [att_name_2...att_name_n] | ALL}
rel_name
[condition_1 [{AND | OR} condition_2...]]
```

   where the attribute clause and condition clause are extracted from the query for the relation involved, and copied to the remote node by a subprocess. The program places itself into hibernation until the process is done, when the subprocess wakes up the program. The query file is created by decomposing a RIM query, as follows:

1. The Oracle database name and password are extracted from the data dictionary.

2. If a data retrieval command is involved, the attribute clause is the union of the attributes from the SELECT/TALLY/COMPUTE clause, the WHERE clause and the SORTED BY clause of the appropriate query. If a project clause is involved, the attribute clause is the union of the attributes from the USING and WHERE clause of the query. If a join or union command is involved, the attribute clause is just all the attributes of the relation involved, extracted from the data dictionary. If a intersect or subtract command is involved, the attribute clause is the common attributes of the two relations involved in the query.

3. If a database retrieval or a project command is involved, the condition clause is just the WHERE clause of the query. For other relational algebra commands, the condition clause is null.

2. The program in the remote node is activated, as described in Section 5.4. When the program in the remote node is done, a result file is copied to the host node. (The remote node program is described in Section 5.5.2.)

3. When the result file is received, the program will retrieve data from it and load the data into a RIM database.

4. Steps 1, 2 and 3 can be repeated if both relations in a query are located in Oracle.

5. When both relations are located in a RIM database, the original query from the user can then be performed, using the method described earlier.

5.5.1.5 Setup Module

This module contains the procedures for performing the SETUP query to setup a data dictionary for each user.

The following is a description of the method used to setup a data dictionary, providing details not described in Section 5.3:

1. If the relation is stored in RIM, its definition is extracted by a LISTREL command to obtain its attributes' name, type and length, using the method described earlier to access RIM.

2. If the relation is stored in Oracle, its definition is extracted by a DESCRIBE command to obtain its attributes' name, type and length. This is done by:

    1. A query file is created in this format:

        2
        Oracle_database_name
        Oracle_database_password
        rel_name

        where rel_name is the relation involved, and copied to the remote node by a subprocess for processing.

    2. When the result file is received, the program retrieves the definition of the relation involved. (The remote node program is described in Section 5.5.2.)

3. A query file is created with RIM commands to define a database and load its relations.

4. The method described earlier to access RIM is used, except that a result file is not produced by RIM.

5.5.1.6 Database Retrieval Module

This module contains procedures that prompts the user for the input for each database retrieval command, and the procedures for creating the query file for each query.

5.5.1.7 Relational Algebra Module

This module contains procedures that prompts the user for the input for each relational algebra command, and the procedures for creating the query file for each query.

5.5.1.8 Login Module

This module contains the procedures for performing the database retrieval and relational algebra commands. The following are simplified pseudo-Pascal descriptions of the procedures for performing database retrieval and relational algebra commands:

```
PROCEDURE database_retrieval;
BEGIN
  REPEAT

    obtain choice of operation;  { Figure 4.6 }
    obtain input for query;      { Figure 4.7 - 4.11 }

    IF relation involved NOT IN TABLES linked list THEN
      error := TRUE;

    IF NOT error THEN
      BEGIN
        IF necessary THEN
          obtain data dictionary information to create
          COLUMNS linked list for relation involved;

        IF attributes involved NOT IN COLUMNS linked
           list THEN
          error := TRUE
      END;

    IF NOT error AND (relation IN Oracle) THEN
      BEGIN
        create query file;

        IF remote node reachable THEN
          BEGIN
            copy query file to remote node;
            start program in remote node;

            IF result file received THEN
              load into RIM
          END
        ELSE
          error := TRUE
      END;

    IF NOT error THEN
      BEGIN
        create query file;
        start subprocess execution on RIM;
        hibernate program;
        display result           { Figure 4.18 }
      END

  UNTIL user is done
END; { procedure database_retrieval }
```

```
PROCEDURE relational_algebra;
BEGIN
  REPEAT

    obtain choice of operation;  { Figure 4.12 }
    obtain input for query;      { Figure 4.13 - 4.17 }

    IF relations involved NOT IN TABLES linked list THEN
      error := TRUE;

    IF NOT error THEN
      BEGIN
        IF necessary THEN
          obtain data dictionary information to create
          COLUMNS linked list for relations involved;

        IF attributes involved NOT IN COLUMNS linked
           list THEN
          error := TRUE
      END;

    WHILE NOT error AND (a relation is IN Oracle) DO
      BEGIN
        create query file;

        IF remote node reachable THEN
          BEGIN
            copy query file to remote node;
            start program in remote node;

            IF result file received THEN
              load into RIM
          END
        ELSE
          error := TRUE
      END;

    IF NOT error THEN
      BEGIN
        create query file;
        start subprocess execution on RIM;
        hibernate program;
        display result           { Figure 4.18 }
      END

  UNTIL user is done
END; { procedure relational_algebra }
```

5.5.1.9 RIM-Oracle Program

This is the main program of the RIM-Oracle interface system in the host node. It contains procedures to perform the setup, login and help functions. The following is a simplified pseudo-Pascal description of the main program:

```
BEGIN { program RIM_Oracle }
  REPEAT
    obtain choice of operation;  { Figure 4.1 }
    CASE choice OF
      '1' : setup;
      '2' : login;
      '3' : help;                { Figure 4.19 }
      '4' : { exit };
      OTHERWISE
    END
  UNTIL choice = '4'
END. { program RIM_Oracle }
```

The following are simplified pseudo-Pascal descriptions of the Setup and Login procedures called by the main program:

```
PROCEDURE setup;
BEGIN

  obtain login information;   { Figure 4.2 }

  REPEAT

    obtain input for query;   { Figure 4.3 }

    IF relation involved IN RIM THEN
      BEGIN
        create query file;
        start subprocess execution on RIM;
        hibernate program;
        extract information from result file
      END

    ELSE IF relation involved IN Oracle THEN
      BEGIN
        create query file;

        IF remote node reachable THEN
          BEGIN
            copy query file to remote node;
            start program in remote node;

            IF result file received THEN
              extract information from result file
          END
        ELSE
          error := TRUE
      END;

    IF NOT error THEN
      BEGIN
        IF necessary THEN
          create new data dictionary;
        add information into data dictionary;

        IF necessary THEN
          add information into TABLES linked list
      END

  UNTIL user is done
END; { procedure setup }
```

```
PROCEDURE login;
BEGIN

  obtain login information and access data dictionary
  to create TABLES linked list;  { Figure 4.4 }

  REPEAT

    obtain choice of operation;  { Figure 4.5 }

    CASE choice OF
      '1' : database_retrieval;

      '2' : relational_algebra;

      '3' : spawn to RIM;

      '4' : spawn to VAX/VMS;

      '5' : setup;

      '6' : { Main Menu };

      OTHERWISE
    END

  UNTIL choice = '6'

END; { procedure login }
```

5.5.2 Remote Node Module and Program

One module and one main program form the remote node portion of the interface system. The remote node program is started by the command procedure activated by the host node program. Its activation means that a query file has been received, and needs to be processed.

5.5.2.1 Oracle Module

This module contains procedures that process the query file from the host node, producing a query file for Oracle, and procedures that modify the result file from Oracle into a format required by the host node program. The following is a description of the method used to perform the above:

1. The query file from the host node can be in one of two formats. If a "1" precedes the database name and password, then the statements after the name and password are inputs to the SELECT command of SQL, described in Chapter 3. If a "2" precedes the database name and password, then the statement after the name and password is the input to the DESCRIBE command of SQL, described in Chapter 3.

2. If a SELECT operation is to be performed, the inputs to this command must be modified from the RIM format to the Oracle format. The input to the SELECT operation is modified as such:

    1. The input to the SELECT clause is modified from att_name_1 [att_name_2...att_name_n] to att_name_1 [,att_name_2,...,att_name_n], and from ALL to *.

    2. The input to the FROM clause need not be modified.

    3. The input to the WHERE clause (if present) is modified as such: the execution of the WHERE clause in RIM is from left to right, whereas the execution of the WHERE clause in Oracle gives precedence to ANDs over ORs. Therefore parentheses are added to the WHERE clause to force Oracle to execute it from left to right. The conditional operators in RIM (EQ, NE, EQA, NEA) are converted to the equivalent conditional operators in Oracle (=, <>). Also the EXISTS and FAILS functions do not exist in an Oracle WHERE clause, and are therefore removed.

3. If a DESCRIBE operation is to be performed, the input need not be modified.

4. After step 2 or 3, a query file in RIM format is modified to a query file in Oracle format, and therefore the method described in Section 5.5.1.3 can be used to obtain a result file from executing the query on Oracle.

5. The result file is modified to the format required by the host node program. For a SELECT operation, the result is a relation, and for a DESCRIBE operation, the result is the definition of a relation. (Actually there is a form of the SELECT command whose result is a relation containing the definition of a relation. But this relation must still be converted into the format required by the host node program.)

    1. FOR a SELECT operation, the result file is modified as shown below:

Result file from Oracle:

```
SQL> SELECT...FROM...[WHERE...];

att_name_1 att_name_2...att_name_n
---------- ----------   ----------
.
.
.

SQL> ^Z
Disconnected from ORACLE
```

Result file after conversion:

```
att_name_1 att_name_2...att_name_n
---------- ----------   ----------
.
.
.
```

2. FOR a DESCRIBE operation, the result file is modified as shown below:

Result file from Oracle:

```
SQL> DESCRIBE rel_name;
 Name          Null?    Type
 ------------ -------- ----
 att_name_1             NUMBER(size, dec)
 att_name_2             NUMBER(size)
 .
 .
 .
 att_name_n             CHAR(size)

SQL> ^Z
Disconnected from ORACLE
```

Result file after conversion:

```
Name         Type Length
------------ ---- ------
att_name_1 REAL size
att_name_2 INT  size
.
.
.
att_name_n TEXT size
```

The following is an example of a query file from the host node, and the query file after conversion for Oracle:

Query file from host node:

```
1
UHDB
UHADMIN
SSN GRADE
TRANSCPT
GRADE EQ "A" OR GRADE EQ "B" AND SSN EQ "123456789"
```

Query file after conversion:

```
UHDB/UHADMIN
SPOOL OUT.DAT
SELECT SSN, GRADE
FROM   TRANSCPT
WHERE  (GRADE = 'A' OR GRADE = 'B') AND SSN = '123456789';
EXIT
```

5.5.2.2 Oracle-RIM Program

This is the main program of the RIM-Oracle interface system in the remote node. The following is a simplified pseudo-Pascal description of the main program:

```
BEGIN { program Oracle_RIM }
  OPEN DECnet connection to host node program;
  access query file from host node to
  create query file for Oracle;
  start subprocess execution on Oracle;
  hibernate program;
  modify result file to format required by
  host node program;
  copy result file to host node;
  CLOSE DECnet connection to host node program
END. { program Oracle_RIM }
```

5.5.3 Host Node and Remote Node Command Procedures

Five command procedures, in the Digital Command Language (DCL) of VMS, are required in the host node and three command procedures are required in the remote node by the interface system [VMS 86(A)]. These command procedures are executed by the subprocesses created by the host node and remote node programs.

The command procedures are for starting the RIM and

Oracle DBMSs for processing of query files; for copying files to and from the host node and remote node; for spawning the user to RIM; and for testing that the DECnet connection between the host node and remote node is functioning. In addition, one command procedure is used for starting the remote node program from the host node, as described earlier.

5.5.4 Error Checking

Extensive error checking is done by the host node and remote node programs. Self-explanatory error messages are displayed whenever an error is encountered, and the user is allowed to continue with another operation.

5.6 Design Decision

The host node and remote node programs are designed to be as independent of each other as possible. Interface between the host node and remote node programs occurs in two instances: (1) when the query file is copied to the remote node, and (2) when the result file is copied to the host node.

The query file that is copied to the remote node is in one of two formats, and the remote node program must recognize that the input from a format 1 query file is in

RIM, and must be converted to SQL; therefore there is dependence here. The result file from Oracle is in a fixed format that is required by the host node program; there is no dependence here.

The host node program is independent of the remote node program, and need not know that the remote DBMS is Oracle. The remote node program, on the other hand, is not independent of the host node program in that it does need to know that a query from the host node is in RIM and must be converted to SQL.

5.7 Design Restriction

The RIM-Oracle interface system does not allow database modification (inserting, deleting and modifying tuples, removing relations, renaming attributes and relations) yet.

Chapter 6

CONCLUSION

The RIM-Oracle interface system allows a user to transparently access Oracle through RIM, where the two DBMSs are located in different nodes connected by a network. The following is a summary, a discussion of the complexity and performance of the system, and a suggestion of future extensions to the system.

6.1 Summary

This thesis develops a transparent database interface between two relational database management systems, located in different computers of a computer network, such that they can be viewed as a heterogeneous distributed database management system. An interface program is build on top of each DBMS, and the two programs communicate over the network to perform their tasks. The host DBMS, RIM, is independent of the remote DBMS, Oracle, whereas the remote DBMS is dependent on the host DBMS, although the dependence is limited.

The third approach to building database interfaces between different DBMSs is used. A host interface module is implemented on top of RIM, and a remote interface module

is implemented on top of Oracle. The methods developed to build the host and remote interface modules can be used to build host-remote pairs of interface modules for different DBMSs of a heterogeneous DDBMS.

6.2 System Complexity and Performance

If there are n different DBMSs, 2n database interface modules are required for this third approach, as opposed to n interface modules required for the second approach of building a common DBMS. The advantage of the third approach over the second approach is the host and remote interface modules are much less complex than the common DBMS, and does not require a global data dictionary. The remote interface module is also much less complex than the host interface module. The RIM host interface module consists of 3600 lines of code, while the Oracle remote interface module consists of 500 lines of code, in Pascal. The methods described in Chapter 5 to build a host and remote interface module are not complex, and can be used to build host-remote pairs of interface modules for other DBMSs.

The performance of the interface system is compared to the direct approach of using command procedures to perform queries on relations located in RIM and Oracle:

Operation		Initial Table Size	Result Table Size	Total CPU Time (s)	Total Elapsed Time (s)
SELECT query on RIM relation	Direct	100	100	3.90	8.42
	System	100	100	5.40	20.56
JOIN query on RIM relations	Direct	100	100	7.35	15.65
	System	100	100	15.88	35.17
SELECT query on Oracle relation	Direct	100	100	6.15	78.08
	System	100	100	23.57	103.34
JOIN query on Oracle relations	Direct	100	100	24.12	105.36
	System	100	100	49.41	192.01
JOIN query on RIM and Oracle relations (System)		100	100	27.62	116.19

Total CPU Time is the total execution time of the program and its subprocesses (in the host and remote nodes).

Total Elapsed Time is the time from when the query is made to the time when the query is executed.

Figure 6.1 Performance of RIM-Oracle Interface System

The following are the queries used in the performance test:

RIM query:

```
SELECT ssn name addr
FROM   student

JOIN     student
USING    ssn
WITH     transcpt
USING    ssn
FORMING studtran
```

(STUDENT and TRANSCPT are located in RIM.)

Oracle query:

```
SELECT secid, ssn, grade
FROM   transcpt;

SELECT *
FROM   student, transcpt
WHERE  student.ssn = transcpt.ssn;
```

(STUDENT and TRANSCPT are located in Oracle.)

Interface System query:

```
JOIN     student
USING    ssn
WITH     transcpt
USING    ssn
FORMING studtran
```

(STUDENT is located in RIM and TRANSCPT is stored in Oracle.)

The performance of the JOIN query on a relation located in RIM and a relation located in Oracle is diagramed in the following:

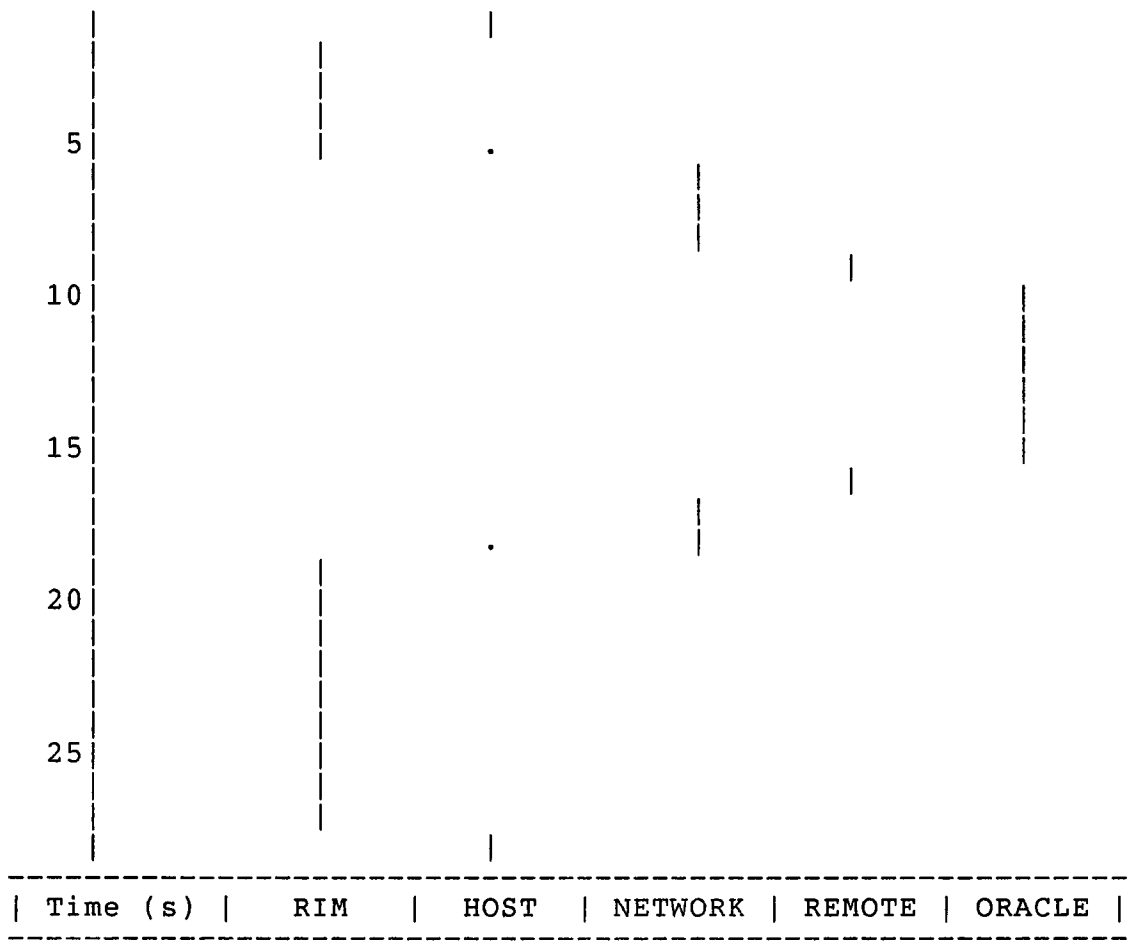


Figure 6.2 Performance of Join query

6.3 Extensions

The following extensions to the RIM-Oracle interface system are envisioned:

1. The design restriction of the interface system, of not allowing database modification, can be removed. The interface system can be extended to allow database modification using the methods described in Chapter 5, thus providing to the user a full range of data manipulation commands.

2. The host node program is independent of the remote node program, and need not know that the remote DBMS is Oracle. The interface system can thus be extended to allow queries involving other relational DBMSs, besides Oracle, located in other nodes of the network, using the methods described in Chapter 5.

3. The methods described can be used to build host-remote pairs of interface modules for each local DBMS of a computer network to create a heterogeneous distributed database management system.

REFERENCES

[CERI 84] Ceri, S., and G. Pelagatti, Distributed Databases: Principles and Systems, McGraw-Hill, 1984.

[DATE 85] Date, C. J., An Introduction to Database Systems, Vol. 1, 4th ed., Addison-Wesley, Reading, MA, 1985.

[JPLRIM 85] Johnson, J., and I. Johnson, RIM User's Guide, JPL Version 4.5, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, February, 1985.

[ORACLE 81] Relational Software Incorporated, Oracle User's Guide, Version 2.3, April, 1981.

[UHRIM 86] University of Houston-Downtown, UHRIM Reference Manual, Version 1.0, Houston, TX, September, 1986.

[ULLMAN 82] Ullman, J. D., Principles of Database Systems, 2nd ed., Computer Science Press, Rockville, MD, 1982.

[VMS 85] Digital Equipment Corporation, VAX/VMS

Programming in VAX Pascal, VAX/VMS Version 4.0, Maynard, MA, March, 1985.

[VMS 86(A)] Digital Equipment Corporation, VAX/VMS DCL Dictionary, VAX/VMS Version 4.4, Maynard, MA, April, 1986.

[VMS 86(B)] Digital Equipment Corporation, VAX/VMS Networking Manual, VAX/VMS Version 4.4, April, 1986.

[VMS 86(C)] Digital Equipment Corporation, VAX/VMS Run-Time Library Routines Reference Manual, VAX/VMS Version 4.4, April, 1986.

[VMS 86(D)] Digital Equipment Corporation, VAX/VMS System Services Reference Manual, VAX/VMS Version 4.4, April, 1986.