RELATIONAL
DATABASE
SYSTEMS, INC.

MARATHON

Relational
Database Management
System

1 What is MARATHON?

Nearly all computer users store information on disks, modify it regularly, and wish to query it and have reports written from it. If the application is very common, such as payroll, a packaged program may be available to perform these tasks. However, most user's requirements have a great deal of uniqueness. For example, a blood bank's supply tracking needs probably won't be satisfied by the capabilities of a ready-made inventory control package. In fact, many companies inventory control requirements are not met by the capabilities of any pre-programmed packages. Some customization is usually needed, and that means programming. If no package is available at all, the amount of programming needed to start from scratch is often prohibitive.

Many consultants, system designers and programmers are turning to UNIX to increase their custom programming productivity. UNIX provides not only a pleasant environment for software development, but also provides a wide variety of programs that are already written, and designed to work together. Programmers are using these off-the-shelf components to build custom systems without starting completely from scratch. By doing this they are cutting their costs and increasing their productivity enormously.

As valuable as UNIX is, however, it lacks the most important tool for building commercial application software, a database management system (DBMS). A database management system is a collection of programs that work together to make the construction of an application system simply a matter of assembling a flexible set of building blocks, using a simple set of tools. The resulting application system is built faster and is more dependable because the building blocks are sturdier than newly-written code. The same tools used to build the software make it easy to modify and enhance as the user's needs mature and change.

MARATHON is a collection of such programs designed to aid the building of single or multi-user applications on UNIX, UNIX look-a-like and UNIX-compatible operating systems. MARATHON's components include an interactive query language, an interactive data entry and maintenance program, several report writers, audit trail and recovery programs and all the utilities needed to create, optimize and modify databases.

This set of tools is so flexible and complete that many simple applications need only be configured, and not programmed at all. For situations where detailed customization is a must, powerful interfaces to standard programming languages are naturally part of the system.

2 The INFORMER Query Language

The INFORMER query language allows users to interactively list any subset of the data in the database that is desired. Consider the two data files below to be part of a MARATHON database maintained by a retail store that operates on credit.

A customer of the store orders items, and when they are delivered the customer is billed, and assumes a non-zero balance until the bill is paid. The first file, **customers**, contains a name, address and balance. There will be only one record in this file for each customer.

The **orders** file contains a record for each order that is pending. The records contain the customer's name, the item ordered and the quantity of the item ordered.

customers

cname	address	balance
Brooks, B.	7 Apple Rd.	10.50
Field, W.	43 Cherry Ln.	0
Robin, R.	12 Heather Ct.	23.45
Hart, W.	65 Lark Rd.	43.00
Court, S.	56 Blossom Rd.	0
English, D.	82 Alpine Rd.	0

orders

oname	item	quantity
Brooks, B.	Work Bench	5
Brooks, B.	Saw	1
Robin, R.	Work Bench	3
Hart, W.	File	3
Robin, R.	Hammer	8
Court, S.	Saw	3
Court, S.	File	5
English, D.	File	1
English, D.	Hammer	2

Perhaps the store manager has determined that the people that have their bills paid at any given time are the people that pay their bills early. He would like a list of their names so he can service them more promptly. The READ and PRINT commands of INFORMER could be typed as below to provide such a list.

```
read into x cname, balance where balance = 0 end
print x end

 cname          balance

 Field, W.         0
 Court, S.         0
 English, D.       0
```

The READ command creates a scratch file, in this case called **x**, that contains the **cname** and **balance** fields from the file CUSTOMERS. However, it contains only the records that reflect a zero balance. Taking a subset of the records in a file is called a SELECTION. Also notice that the file **x** has only two of the three fields that are in CUSTOMERS. Taking a subset of the fields is called a PROJECTION.

SELECTION and PROJECTION are two of the three relational operators that make MARATHON a relationally complete DBMS. The third operator, the JOIN, allows questions to be answered based on information that exists in more than one file. For example, now that the store manager has the names of the people who pay their bills, he may want to know what they are ordering so he can be careful to keep these items in stock. This requires that **item** fields of the **orders** file be printed based upon the contents of the **balance** field of the **customers** file.

The **customers** file and the **orders** file are related. The **customers** file contains a field that holds customer's names, **cname**. The **oname** field of the **orders** file contains some of the same names. The relationship is called a one to n mapping. For each record in the **customers** file, there could be n records in the **orders** file.

The JOIN operator can be used to exploit this relationship and the fields that link the files together. The scratch file **x** already contains the names of the customers with their bills paid. The following READ command will pull the items out of the **orders** file that are on order by the customers in file **x**. These items will be put in another scratch file, this one called **y**.

```
read into y item from join on x.cname = oname end
print y end

        item

        File
        Hammer
        Saw
```

If there are other files in the database, the JOIN operator can be used again to easily answer questions that are even more involved. There is no limit to the number of files that can be used to answer a question. The only requirement is that the fields involved in a JOIN be key fields. The interactive database utility DBSTATUS can be used to turn a non-key field into a key field at any time.

3 ENTER I Turn-Key Data Entry Program

The data in the **customers** and **orders** files got there by one of two means, a custom application program or the ENTER I automatic data entry program.

ENTER I has interactive commands that allow the user to add, delete and update records. Records can also be found based on the contents of a field. If that field is a key field, the next or previous record in sorted order can be found.

In the example below, the **customers** record for **Brooks** is added. All of the commands can be abbreviated with their first character.

```
> a

cname
>> Brooks, B.

address
>> 7 Apple Rd.

balance
>> 10.50

cname          Brooks, B.
address        7 Apple Rd.
balance        10.50
```

If any of the information is incorrect, the update command can be used to change it.

4 Application Language Library

The subroutines that are used by the INFORMER query language and the ENTER I data entry program are available to the database programmer as a relocatable library. This small collection of parameterized calls turns the C programming language into a relational database programming language. Interfaces to other languages will also be available.

The routines make the database capabilities available at a very high level. For example, the DBREAD routine allows the programmer to perform any operations that can be performed interactively with the READ command of INFORMER, with only one subroutine call. DBASSIGN makes all of the set operations of INFORMER available.

The records in database files can be read, written or deleted with the DBGET, DBADD and DBDELETE calls. Routines are also available to load a permanent file from a temporary file, and erase a permanent file. DBADDKEY and DBREMOVEKEY can be used to optimize database performance or make a field a key when uniqueness or a sorted ordering is required. Concurrency control is provided by DBLOCK and DBUNLOCK.

The MARATHON manuals discuss how these calls can be used together to easily create custom query languages or data entry programs with a minimum of programming.

5 Full Data Integrity for Multiple Users

Interactive database modification in a multi-user environment has always posed a tricky technical problem known as concurrency control. The DBLOCK and DBUNLOCK routines in the Application Language Library allow programs to be written that eliminate this problem. ENTER I uses these routines so that MARATHON users need never worry about loss of data integrity due to problems with concurrency control. The MARATHON manuals contain a tutorial on this problem which shows how ENTER I uses these calls to solve the problem while letting a maximum amount of concurrent update activity take place.

6 Audit Trail Backup and Recovery

Often when an application software system is built the problems of backup and recovery are put off until the end, and sometimes forgotten about completely. The daily or weekly backup of the file system onto tape is only one part of a useful backup and recovery program.

If the data on the disk is lost due to hardware or software failure, it is only recoverable back to the last time a backup was made. A great many transactions can still be lost, and users will be forced to maintain paper backup systems, and depend upon them. This may significantly lessen the benefits of the automated system.

The MARATHON program DBSTATUS can be used to activate an automatic transaction audit trail. This audit trail can be kept on any desired disk or tape. It is recommended that the physical device be different than the device that holds the data.

If a failure should occur, the data can be recovered from the backup tapes. Then all of the transactions that were performed since the backup can be re-run against the database programmatically, to restore the system completely to the time of failure.

Some of the transactions that were recorded in the audit trail may not be desired in the restored database. Perhaps they were associated with the program that caused the damage. Selected transactions can be kept out of the restoration based upon the time of the transaction, the process that executed it, the user that initially ran the program, the content of the transaction itself or any combination.

7 MARATHON Report Writers

When buried in the complexity of building a software system by conventional means it is easy to overlook the fact that people only put information into a computer so they can get it back out. Database industry observers have predicted that report writers will be the "tail that wags the dog." No single feature of a database system improves its usability as much as a good report writing language.

Most commercial database systems provide one report writing language. Unfortunately, this language often falls short of what is needed to produce a wide range of commercial reports. The least effective languages are only extended query languages which do minimal formatting or calculation.

RDS realizes the importance of the report writing function. We have chosen to leave our query language simple, and attack the report writing problem with several english-like languages that provide the calculation and formatting features needed in the commercial world.

ACE is a general-purpose report writer, useful in creating standard data processing reports with page headings, column headings and columns of data sorted in any order. The functions TOTAL, AVERAGE, PERCENT and others are available. ACE is designed to meet the needs of the reporting problems it confronts with less than one tenth of the programming effort that would be needed to write equivalent programs in a conventional language.

The ten-fold increase in productivity is possible because ACE is designed to be a tool to perform a restricted class of tasks, write reports. Because this is assumed, the language only contains syntax to state what the report is to look like, not how the files are to be opened, the data read, calculated, sorted and written. ACE takes care of all the details of the algorithm of the program. Languages such as ACE that do not require the programmer to break the task down into subtasks and algorithms are known as non-procedural languages, or specification languages.

The MARATHON system will constantly grow with the addition of new report writers. All of them will be non-procedural and will address a particular class of reporting needs. They will all contain the same constructs as the INFORMER query language for selecting the data from the database.

The PERFORM report writing language is designed to fill out pre-printed forms, such as government or insurance forms, with data drawn from the database. PERFORM is an excellent tool for building application systems that are form intensive, such as medical or legal office management systems.

DBMAILER is designed to print mailing labels or type addresses directly onto envelopes. Combining this formatting ability with the database selection power of MARATHON will produce the world's best mailing label package.

INFER will be a language designed specifically for scientific labs, and will greatly aid in the production of reports that perform inferential statistics. Once again, combined with the MARATHON system, this statistics package will be far more usuable than those that make the user worry about how to get the data into the computer and manage it while it is there. The relational model also allows researchers to model their data so that sophisticated questions, possibly involving data in many files, can be answered easily.

8 Sophisticated Internal Architecture Provides High Performance

There are non-relational models for storing and retrieving information with computers. Systems based on non-relational models have been justifiably praised. Considering the alternative has been to start building an application from scratch, they have been enormously useful tools. These models evolved from internal programming techniques that were easy to implement and ran quickly in the computer.

Unfortunately, much of this software that is convenient for the computer to execute has proven to be inconvenient for application programmers to use. Non-relational systems are notorious for being difficult to use in arbitrary ways, due to artifacts of how the software evolved. The databases built cannot always be queried on an ad hoc basis. New reports often require database restructuring. Finally, most systems lack the utilities to perform any restructuring easily.

The relational model was invented and refined during the 1970's. During that time many prototype systems were built in the academic community as research projects. Some research work was also done commercially. This work was aimed at defining a set of data structures and operations that performed a great deal of desired work simply, while restricting the user as little as possible. The result was the relational model.

The public was told that although this new model was wonderful for the user, it was difficult to implement with adequate performance. Indeed, implementations of these early systems did not perform well, and for the most part, still don't. However the reasons for the poor performance have been recognized and alternative methods realized.

MARATHON benefits greatly from this research work and has eliminated performance problems by heeding these lessons. MARATHON has been kept sleek, and runs as one process on UNIX. The b-tree technique of building key files is used, and is the only method that is used. MARATHON rests on top of the operating system, and does not require any changes to perform well. MARATHON was designed to be a multi-user system from the beginning. The system has not had to be altered or performance compromised due to internal architecture ill-suited for multi-processing.

9 Summary

MARATHON is the first commercial DBMS for UNIX. It is not based upon any particular academic system or commercial prototype, but rather it draws from the best features of all of them. The sole reason for its existence is the commercial UNIX user, and the design and future directions of the product reflect this.

Good documentation, error messages, enhancements, host language compatability, training and support are all expected by commercial users. RDS is committed to helping system designers and programmers cut the costs and heighten the reliability and expandability of their applications. The MARATHON system and its report writers are the base for that help.

No application should be built for UNIX starting from scratch. Any data intensive software system should include the data manipulation and maintenance functions provided by MARATHON. Systems designers and applications programmers owe it to their users to start building an application with a firm foundation.

For a license agreement, product availability bulletin or order form call or write to:

Relational Database Systems, Inc.
1208 Apollo Way
Suite 503
Sunnyvale, California
94086

(408) 746-0982

RELATIONAL
DATABASE
SYSTEMS, INC.

1208 Apollo Way, Suite 503
Sunnyvale, California 94086
Telephone 408/746 0982